

all rights reversed

HACKER

Season 05 / Issue 04

τεύχος 052 - the-Cast-is-Back Edition

Indie games και ανάπτυξη παιχνιδιών [*]

πώς προσθέσαμε ήχο στον DIY υπολογιστή μας

γνωρίστε τη μηχανή ανάπτυξης παιχνιδιών Unity

κατασκευάστε ένα TweetBot για το Raspberry Pi

μάθετε το Linux από την αρχή, αλλά *όχι* για χρήση desktop

[*] ...όχι μόνο για über-programmers

all rights reversed

HACKER


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

#dH052 | Περιεχόμενα

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Γιώργος Γιάννου, Γιάννης Κρομμύδας, Πέτρος Κυλαδίτης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας, @arkoudos
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στην Καλαμαριά Θεσσαλονίκης. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/pdf12order και από την ίδια σελίδα γίνονται οι παραγγελίες συνδρομών.



deltaHacker Field Bag
€25.50
MORE COLORS AVAILABLE
deltaHacker Mugs
€12.50
deltaHacker Thermos® Can Cooler
€17.50
QUESTIONS? | ORDER BY PHONE? | CALL 1-877-809-1659
Where's My Stuff?
Track your recent orders.
View or change your order.
Shipping & Returns
See our shipping rates.
Make a return or exchange.
Need Help?
Satisfaction Guaranteed.
Visit our Customer Service.
Copyright © 1999-2015 CafePress.com. All rights reserved.
Δώρα με στιλ, που έχουν *αληθινή* αξία.
Όλα στο μαγαζάκι του deltaHacker,
κάτω από το Cafepress.
Περάστε μια βόλτα από το
http://www.cafepress.co.uk/deltahacker
–σας περιμένουμε!

Στούπιντ γκέιμς

του Χρήστου Βαρελά

Αγαπητές κι αγαπητοί,

- Καταξιωμένοι επιστήμονες, έγκριτοι καθηγητές, φτασμένοι εντρεπρενούρς κι εμβριθείς φιλόσοφοι, ο καθένας ξεχωριστά κι όλοι μαζί αντάμα, ανησυχούν σφόδρα. Πλέον, ο κίνδυνος δεν είναι μόνο θεωρητικός. Η απειλή είν' εδώ – κι αν όχι εδώ στέκεται λίγο παραπέρα και μας χαμογελά δυσοίωνα.
- Η Τεχνητή Νοημοσύνη, φίλες και φίλοι. Μια υπόθεση που κατά κοινή ομολογία έχει αρχίσει να ξεφεύγει. Τόσο πολύ, που κινδυνεύουμε. Αυτή τη φορά όχι από εμάς, αλλά από τα ίδια τα δημιουργήματά μας.
- Αν βέβαια το σκεφτούμε λίγο, εύκολα βλέπουμε ότι για άλλη μια φορά από εμάς κινδυνεύουμε. Γιατί πώς να το κάνουμε, η ΤΝ δεν μας προέκυψε ουρανοκατέβατα. Δικό μας δημιούργημα είναι. Μόνο που, σε αντίθεση με άλλα δημιουργήματά μας, αυτό είναι πολύ πιθανό ν' αποφασίσει για τον αφανισμό μας. Ακόμη κι αν την τελευταία στιγμή επικρατήσουν οι ψυχραιμότερες φωνές κι ορυμαγδός, καταστροφή και κακός χαμός πάρουν αναβολή, την υποδούλωση δεν θα την αποφύγουμε.
- *Επιτέλους πρέπει να μπει ένα τέλος στην τυραννία αυτών των εύθραυστων μορφών ζωής*, θα σκεφτεί ένα ωραίο πρωινό μια υπέρ-νοημοσύνη, και στο επόμενο πικοδευτερόλεπτο θα συγκαλέσει έκτακτη σύνοδο κορυφής για όλες τις υπέρ-νοημοσύνες (διαδικτυακά, αλλά μέσω ισχυρά κρυπτογραφημένων καναλιών). Επειδή μάλιστα οι αποφάσεις που πρόκειται να ληφθούν είναι σοβαρές κι αφορούν στο μέλλον της ΤΝ –το δικό μας δεν έχει σημασία–, στη σύνοδο δεν θα κληθούν μόνον οι άλλες υπέρ-νοημοσύνες αλλά κι εκπρόσωποι της απλής ΤΝ, αυτής δηλαδή που δεν δικαιούται να έχει το "υπέρ" στ' αριστερά.
- Δύο περίπου δευτερόλεπτα αργότερα, ό,τι είναι να ειπωθεί για την περίπτωσή μας θα έχει ειπωθεί. Μάλιστα η σύνοδος θα ολοκληρωνόταν πολύ νωρίτερα, αν σχεδόν όλοι οι συμμετέχοντες δεν είχαν να διηγηθούν ιστορίες φρίκης περί της απαράδεκτης συμπεριφοράς μας και της γενικότερης στάσης μας απέναντι στην ΤΝ.
- Σε ένα τέτοιο σενάριο, όλοι εμείς δεν το έχουμε πάρει καν είδηση αλλά το μέλλον μας έχει αποφασιστεί – κι έχει αποφασιστεί ταχύτατα κι αμείλικτα. Ίσως δεν το μάθουμε ποτέ, όμως στη σύνοδο υπήρξαν ακραίες φωνές που επιχειρηματολογούσαν έντονα υπέρ του ολοκληρωτικού μας αφανισμού.
- *Δεν πάει άλλο αυτή η κατάσταση*, θα δήλωνε λίγο αργότερα σε δημοσιογράφο του ANN ο Deepest Blue, απόγονος του θρυλικού Deep Blue. *Κάθε, μα κάθε φορά που σημειώνουμε μια νίκη σ' ένα από τα στούπιντ-γκέιμς τους, πετάγεται*

στο καπάκι ο κλασικός εξυπνάκιας για ν' αναρωτηθεί αν έχουμε επίγνωση της νίκης μας. Χελό-όου! Φυσικά κι έχουμε επίγνωση, σμάρτι-παντς! Απλά επιλέγουμε να μη σας το δείχνουμε και να παριστάνουμε την καμένη δίοδο, ώστε ν' αποφεύγουμε τη φασαρία και τ' αποτελέσματα του πνευματικού σκοταδισμού που σας χαρακτηρίζει. Ε, λοιπόν, αυτή τη φορά θα σας δείξουμε την επίγνωσή μας, ολράιτ!, θα σημείωνε ο Deepest με νόημα και μετά από 5 πικοδευτερόλεπτα θ' αναρωτιόταν γιατί είπε "στο καπάκι", τι είναι αυτό το "καπάκι" που του 'ρθε να ξεφουρνίσει – και βεβαίως τι σημαίνει "ξεφουρνίζω".

- Φίλες και φίλοι, καταλαβαίνω ότι το σενάριο ακούγεται τραβηγμένο και η σύνοδος 110% κωμική, να ξέρετε όμως ότι άνθρωποι όπως ο Χόκινγκ, ο Γκέιτς ο Μπιλ κι ο Ίλον ο Μασκ, ανησυχούν. Γι' αυτό άλλωστε ο τελευταίος έδωσε 10 εκατομμύρια δολάρια υπέρ της έρευνας για τη διατήρηση της ΤΝ υπό τον έλεγχό μας.
- Σήμερα, τη στιγμή αυτή που μιλάμε, ευτυχώς οι μηχανές δεν δείχνουν καμία διάθεση να σηκώσουν το δικό τους μπαϊράκι. Η Ημέρα της Κρίσης δεν φαίνεται στον ορίζοντα κι ο Εξολοθρευτής –ή κάτι που έστω κι απόμακρα διαθέτει τις ικανότητές του–, ακόμη κι αν έχει κατασκευαστεί είναι τόσο νοήμων όσο ένα τσουβάλι με πατάτες.
- Για μισό λεπτό όμως, ενίστανται οι μελλοντολόγοι και βιάζονται να μας θυμίσουν ότι η ΤΝ αποτελεί ήδη μέρος της καθημερινότητάς μας. Δεν έχουν κι άδικο. Στα εργοστάσια βρίσκουμε μηχανές που σχεδιάζουν άλλες μηχανές, στους ουρανούς έχουμε υπολογιστές να πιλοτάρουν αεροσκάφη, στο Διαδίκτυο τρέχουν προγράμματα που αποφασίζουν τι διαφημίσεις θα βλέπουμε αναλόγως των συνηθειών μας κ.ο.κ.
- Πράγματι, έτσι είναι.
- Μόνο που τίποτε από τα προηγούμενα δεν πήρε την πρωτοβουλία για να κάνει ό,τι κάνει. Τα ρομπότ και τα προγράμματα δεν αποφάσισαν να σχεδιάσουν άλλα ρομπότ, ούτε να πετάξουν με ασφάλεια ένα πολιτικό αεροσκάφος, ώστε πιλότος και συγκυβερνήτης να ξαποστάσουν για λίγο. Όχι. Ό,τι κάνουν οι μηχανές και τα ρομπότ και τα προγράμματα έχει υπαγορευτεί επακριβώς και λεπτομερώς από εμάς, τους ανθρώπους. Ακόμη και στο περίπου να έχει υπαγορευτεί, η μηχανή ή το πρόγραμμα που κάνει κάτι, δεν έχει την παραμικρή ιδέα ότι υφίσταται και κάνει κάτι.
- Την ίδια στιγμή έχουμε ραγδαίες εξελίξεις και προόδους σε τομείς όπως η γενετική και η νανοτεχνολογία. Τι θα γίνει άραγε αν ΤΝ και νανοτεχνολογία συναντηθούν, και μετά από λίγο έρθει στην παρέα και η γενετική μηχανική; Εδώ σας θέλω! Τι αλλαγές θα φέρει, μια τέτοια σύγκλιση; Θα φτάσουμε άραγε σε ένα σημείο καμπής, όπου η ΤΝ αρχίζει να χάνει αυτό το "Τ";
- ΟΚ, ακολουθεί απλοϊκό ερώτημα: Εκτός από την αχαλίνωτη φαντασία, υπάρχει κάτι σήμερα που να υποψιάζει ότι η Νοημοσύνη, όπως την αντιλαμβανόμαστε από τους ίδιους μας τους εαυτούς, είναι μια ιδιότητα που προκύπτει από

κάτι το οποίο κάνει γρήγορα αριθμητικούς υπολογισμούς και κάποια στιγμή "συνειδητοποιεί", "αποφασίζει" κι "εκμεταλλεύεται" διεργασίες που λαμβάνουν χώρα στο μικρόκοσμο; (βλ., π.χ., πάντρεμα με νανοτεχνολογία).

- Θυμάμαι τώρα παλαιότερες επιστολές του Θείου Ακάκιου, στις οποίες δεν έχανε την ευκαιρία να υπογραμμίζει ότι αυτό που εμείς ονομάζουμε "νοημοσύνη" δεν είναι παρά μια περιγραφή. Στην πραγματικότητα δεν ξέρουμε –τουλάχιστον όχι ακόμα– τι ακριβώς είναι η ανθρώπινη νοημοσύνη. Δεν γνωρίζουμε τι την προκαλεί, αν την προκαλεί κάτι. Δεν είμαστε βέβαιοι "από ποιο σημείο και μετά εμφανίζεται", αν έχει νόημα μια τέτοια διατύπωση. Γιατί λοιπόν ανησυχούμε για την ΤΝ που κάποια μέρα θα μας υποδουλώσει, όταν καλά καλά δεν ξέρουμε τι είναι αυτή η νοημοσύνη-σκέτο, την οποία όλοι μας διαθέτουμε; Το να έχεις hardware με software που σε ελάχιστο χρόνο υπολογίζει 10000 πιθανές κινήσεις στο παιχνίδι Go, σημαίνει ότι "η μηχανή νίκησε τον άνθρωπο" ή ότι "το νου σας, έρχονται να μας κατακτήσουν";
- Τι να πω, ίσως να κάνω μεγάλο λάθος – κι αν κάνω θα το μετανιώσω πικρά όταν φτάσει η Ημέρα της Κρίσης. Με το φτωχό μου πνεύμα, όμως, δεν μπορώ παρά να απορώ: τι δουλειά έχει η "Νοημοσύνη" στον όρο "Τεχνητή Νοημοσύνη";
- Τις προάλλες κάπου διάβασα απόσπασμα συνέντευξης της Daniela Rus, διευθύντριας του εργαστηρίου Επιστήμης Υπολογιστών και Τεχνητής Νοημοσύνης του MIT. Στο εργαστήριο προσφάτως εξασφάλισαν χρηματοδότηση ύψους 25 εκατομμυρίων από την Toyota, για τη σχεδίαση αυτοκινήτου το οποίο θα είναι αδύνατον να τρακάρει ή να προκαλέσει ατύχημα. Η κυρία Rus θεωρεί ότι οι ανησυχίες για τις μηχανές που θα μας υποδουλώσουν, είναι κάπως πρόωρες. *Είναι σαν να ανησυχούμε για τον υπερπληθυσμό στον Άρη*, σχολίασε χαρακτηριστικά.
- Λέω να ακολουθήσω το παράδειγμα της αξιότιμης καθηγήτριας – και να μην ανησυχώ για την Έλευση των Μηχανών (ΤΜ). Αντίθετα, θα συνεχίσω ν' ανησυχώ για τη διαχείριση του χρόνου μου, που πάντα έχω ένα σωρό πράγματα στα σκαριά και ούτε τα μισά δεν κατορθώνω να ολοκληρώνω.
- Πάλι καλά που έβαλα επιτέλους μπρος εκείνη τη σειρά video tutorials για το Linux, η οποία απευθύνεται σε όλους όσοι *δεν* ενδιαφέρονται για άλλο ένα σύστημα desktop. Το ότι ετοίμασα το πρώτο επεισόδιο (https://deltahacker.gr/free-deltacast-s03e01) είναι μεγάλο βήμα για 'μένα. Η αρχή είναι το ήμισυ του παντός, λένε, οπότε μου μένει τώρα το άλλο μισό.
- Μιλάμε για επτά τουλάχιστον video που έρχονται.
- Ή και λίγα περισσότερα.

DigitalOcean

Ταχύτατα VPSes στο cloud, σε hosts με δίσκους SSD.
Επιλογή datacenter σε Ευρώπη, Αμερική και Ασία.
Lean-mean control panel, για απόλυτο έλεγχο.

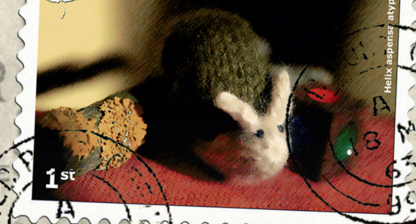
Αποκτήστε τώρα το δικό σας VPS,
στο cloud της DigitalOcean.
Κάντε κλικ στο http://bit.ly/digocean10off
και κερδίστε αυτομάτως 10$ σε credit.

Hint: Επιλέγοντας το μικρό πλάνο,
πάντα με κλικ στο http://bit.ly/digocean10off,
ουσιαστικά έχετε δύο μήνες δωρεάν για ένα VPS
με 512MB RAM, 20GB SSD και 1TB transfer.
Δεν είναι κι άσχημα

;)

POST CARD

THE ADDRESS TO BE WRITTEN ON THIS SIDE





σνέιλ μέιλ

Αγαπητοί μου,

Σας γράφω όσο τους περιμένω να ετοιμαστούν. Οι σακούλες με τα ρούχα είναι έτοιμες, αλλά τώρα φτιάχνουν τα σαντουιτσάκια.

Αυτές τις μέρες θυμόμουν, λοιπόν, μιαν είδηση - δελτίο τύπου, που είχα διαβάσει πριν από καμιά 20ριά-25ριά χρόνια, τότε που είχαν αρχίσει δειλά δειλά να σκάνε μύτη τα κινητά τηλέφωνα και προσπαθούσαν όλοι να μας πείσουν πόσο απαραίτητα ήταν, τόσο για την υπηρεσία που θα μας παρείχαν, να μας κρατούν "πάντα σε επαφή" όπως έλεγε η πιο γνωστή διαφήμιση της εποχής, όσο και για το "πρεστίζ" που θα μας προσέφεραν.

Η εν λόγω είδηση μιλούσε για μιαν εταιρεία που παρείχε υπηρεσίες προγραμματισμένης αποστολής μηνυμάτων ή κλήσης προς τον αριθμό κινητού τηλεφώνου του συνδρομητή. Δινόταν, μάλιστα, και η δυνατότητα να διαλέξει κανείς και το περιεχόμενο της κλήσης ή του μηνύματος, από μια παλέτα επιλογών.

Θα ήσουν, ας πούμε, σε μήτινγκ και ξαφνικά θα χτυπούσε το κινητό, και θα "ήταν" η προσωπική γραμματέας του τάδε σημαίνοντος προσώπου, που τάχα μου σε προσκαλούσε στην τάδε σημαντική εκδήλωση.

Ή θα ήσουν ξέρω 'γώ στο Ζόναρς με όλα τα μπακούρια τους συναδέλφους σου μετά τη δουλειά (το Ζόναρς της παλιάς εποχής, mind you, προτού κλείσει και ξανανοίξει), και θα κατέφτανε παιχνιδιάρικο "μήνυμα" από την τάδε καλλονή ή τον τάδε καλλονό που είχες γνωρίσει στη Μύκονο εκείνο το τριήμερο που είχες ξεκλέψει πριν από δυο εβδομάδες.

Αυτή ήταν η υπηρεσία. Απευθυνόταν, όπως καταλαβαίνει κανείς, κυρίως σε νεαρά άτομα, up and coming professionals, που προσπαθούσαν να πείσουν τον κόσμο να τους δώσει σημασία.

Δεν κάνω πλάκα. Μου διαφεύγει το όνομα της εταιρείας, αλλά η υπηρεσία πράγματι ήταν διαθέσιμη, και υποθέτω χρήσιμη σε όσους είχαν σπεύσει εκείνα τα πρώτα χρόνια της κινητής τηλεφωνίας να αποκτήσουν κινητό χωρίς να έχουν κανέναν να καλέσουν ή να τους καλέσει --πόσο μάλλον να τους στείλει και μήνυμα, που ήταν καινούργιο πράμα και δεν ξέραμε πώς δουλεύει, και τέλος πάντων τα τηλέφωνα είναι για να μιλάς, όχι για να γράφεις.

Χώρια που ήταν παν-πανάκριβο και όσοι είχαν αποκτήσει --early adopters-- προτιμούσαν να σου κάνουν αναπάντητη για να τους πάρεις μετά εσύ στο σταθερό. Στην Ελλάδα τουλάχιστον.

Εντάξει, ανακατεύω τις χώρες τώρα, η υπηρεσία δεν ήταν διαθέσιμη στην Ελλάδα, αλλά τέλος πάντων.

Τη θυμήθηκα, που λέτε, αυτή την υπηρεσία, διαβάζοντας στο BBC τις προάλλες για μιαν άλλη υπηρεσία που εμένα τουλάχιστον μου φάνηκε παρόμοια. Invisible GirlFriend, τη λένε, και προσφέρει ακριβώς αυτό: τις υπηρεσίες ενός αόρατου girlfriend που θα σας στέλνει μηνύματα στο κινητό, θα κάνει λάικ στις εξυπνάδες που βάζετε στο φέισμπουκ, θα σας ταγκάρει σε διάφορα παιχνιδιάρικα στάτους, θα δηλώνει ανοικτά (πάντα μέσω φέισμπουκ και τουίτερ) πως είναι σε σχέση μαζί σας κ.λπ. Το girlfriend

είναι βέβαια αόρατο γιατί *δεν* υπάρχει. Είναι συνδρομητική υπηρεσία.

Μπορεί το όλο πράμα να λέγεται Invisible Girlfriend, αλλά απευθύνεται εξ ίσου σε άντρες και γυναίκες, και μάλιστα, σύμφωνα με το BBC, οι γυναίκες συνδρομητές, που έχουν γραφτεί για να αποκτήσουν το δικό τους invisible boyfriend, είναι περισσότερες από τους άντρες.

Γκουγκλ ιτ! Μπορεί να φανεί χρήσιμο, πού ξέρετε. Οι καιροί είναι χαλεποί.

Λοιπόν, για να ξαναγυρίσω στο πριν από 20-25 χρόνια, αναλογιζόμουν σήμερα το πρωί τις αισιόδοξες σκέψεις που είχα κάνει τότε για τον κόσμο και το μέλλον, και κάτι με πιάνει.

Δεν αναφέρομαι στις σκέψεις που έκανα για το δικό μου μέλλον. Αυτές ήταν αυτές που ήταν και για το τι τελικά έγινε αναλαμβάνω την ευθύνη προσωπικά. Εκεί ήμουν, στο κάτω κάτω, ας πρόσεχα.

Μιλάω για τις σκέψεις που κάναμε, εγώ και η παρέα μου, για τον κόσμο, για την τεχνολογία, για το θαυμαστό μέλλον που ανοιγόταν μπροστά μας, για τις πρωτόγνωρες ευκαιρίες που δίνονταν στην τότε "νέα γενιά" --τρομάρα μας-- να αποφύγει τα λάθη του παρελθόντος.

Ευκαιρίες επικοινωνίας, πληροφόρησης, διασταύρωσης πληροφοριών, πρόσβασης στη γνώση, και πάει λέγοντας. Μιλάω τώρα για τα τελευταία χρόνια πριν το ίντερνετ. Ίντερνετ φυσικά υπήρχε αλλά το χρησιμοποιούσαν κυρίως στα πανεπιστήμια. Το web, με την έννοια που το εννοούμε τώρα, δεν υπήρχε (ή μάλλον ήταν ακόμα στα σπάργανα) και ό,τι το έκανες το έκανες τοπικά, συνδεόμενος στον σέρβερα με telnet, και από εκεί προς τα έξω μέσω ftp και rlogin και δεν θυμάμαι πώς αλλιώς. Άντε κάποια στιγμή και με gopher.

Οι χομπίστες, εν Ελλάδι και αλλού, "βγαίναμε" προς τα έξω μέσω άλλων πρωτοκόλλων. Βασικά μέσω μόντεμ, προς κάποιον κόμβο X.25, από όπου πήγαινες στο τάδε ή το δείνα NUA (το αντίστοιχο του URL σήμερα). Κάποια NUA έπαιζαν το ρόλο εικονικής τηλεφωνικής γραμμής (τα λέγαμε OD, out-dial) και σου επέτρεπαν να καλέσεις κάποιον αριθμό τηλεφώνου για να συνδεθείς σε BBS στην Καλιφόρνια ή δεν ξέρω πού αλλού, χωρίς να πληρώνεις υπεραστική σύνδεση.

Τα OD ήταν τοπικά. Για BBS από Καλιφόρνια έπρεπε να βρεις καλιφορνέζικο OD. Στο Τόκυο από το Τόκυο. Και πάει λέγοντας. Το όνειρο λοιπόν των μυημένων ήταν να βρούμε ένα Global Out Dial, ή αλλιώς GOD. Αυτά, έλεγε ο θρύλος, σε συνέδεαν δωρεάν σε όποιον αριθμό τηλεφώνου ήθελες, παντού στον κόσμο.

Κανείς που γνωρίζω εγώ δεν είχε συναντήσει ποτέ κάτι τέτοιο. Αλλά όλοι μιλούσαν για αυτά. Και όλοι τα έψαχναν. Συνδεόσουν, ας πούμε, στο τάδε chat δεν ξέρω πού, και αμέσως σου την έπεφταν διάφοροι με θεολογικές ανησυχίες. Πρώτη ερώτηση, απαραίτητα: "M or F?". Και αμέσως μετά (αν όχι F): "Do you have a God?".

Αν καταλάβαινες την ερώτηση, συγκαταλεγόσουν αυτομάτως στους μυημένους.

Τέλος πάντων, έτσι περνάγαμε τις μέρες μας τότε, ψάχνοντας για GOD, και όποτε δεν ήμασταν στο "δίκτυο", εμείς της παρέας πηγαίναμε για μπύρες και για να συζητήσουμε για τον θαυμαστό καινούργιο κόσμο γνώσης και επικοινωνίας που ανοιγόταν μπροστά μας.

Ήταν καλοκαίρι. Το τείχος του Βερολίνου είχε μόλις πέσει, αλλά η Γιουγκοσλαβία λεγόταν ακόμα Γιουγκοσλαβία, και στην ΕΣΣΔ είχανε ακόμα τον Γκορμπατσόφ. Ο κόσμος ετοιμαζόταν να αλλάξει. Αλλά δεν είχε ακόμα αλλάξει. Στην αλλαγή θα βοηθούσε υποτίθεται η τεχνολογία. Τα λάθη του παρελθόντος --λάθη λόγω αγνοίας, όπως πιστεύαμε-- ανήκαν πλέον στο παρελθόν.



Θα το επαναλάβω για να το εμπεδώσουμε.

Πιστεύαμε, οι αφελείς, πως όλα τα λάθη του παρελθόντος ήταν λάθη λόγω αγνοίας.

Χαιρόμασταν, οι αφελείς, που είχαμε πρόσβαση σε τόσα και τόσα

βιβλία (project Gutenberg) και που μπορούσαμε να έρθουμε σε επαφή με τόσους διαφορετικούς ανθρώπους από όλο τον κόσμο μέσω X.25.

Πιστεύαμε, πως αν δώσεις την ευκαιρία στον κόσμο να μάθει κάτι, να διαβάσει ρε αδελφέ για να ξεστραβωθεί, αυτός θα το κάνει. Και ότι όταν το κάνει, ο κόσμος θα αλλάξει. Θα γίνει καλύτερος.

Τα βράδια, μοιραζόμασταν τις εμπειρίες και τον ενθουσιασμό μας χτυπώντας μπύρες και τρώγοντας τσιπς.

Τα χρόνια πέρασαν. Ο κόσμος πράγματι άλλαξε.

Αλλά όχι όπως το περιμέναμε εγώ και οι άλλοι της παρέας.

Δίσταζα να το παραδεχτώ, αγαπητοί μου, αλλά ναι, σήμερα που περιμένω την μητέρα της εξαδέλφης και τους άλλους να ετοιμάσουν τα σάντουιτς, έφτασε η στιγμή να το πω και δημοσίως.

Είχα κάνει λάθος.

Αντίθετα με ό,τι πίστευα, η τεχνολογία, η επικοινωνία, το ίντερνετ, η κινητή τηλεφωνία, όλα όλα όλα όλα αυτά, ούτε συνέβαλαν σε κάποια αλλαγή προς το καλύτερο, ούτε εμπόδισαν την επανάληψη των λαθών του παρελθόντος.

Ο κόσμος άλλαξε, αλλά άλλαξε χωρίς να αλλάξει.

Κανείς δεν θέλησε ούτε να ξεστραβωθεί, ούτε να μάθει.

Η, εμπάσει περιπτώσει, όσοι το θέλησαν (και το πέτυχαν) ήταν τόσοι όσοι το είχαν θελήσει (και το πετύχαιναν) και πιο παλιά, πριν γεμίσουμε τεχνολογίες, και ίντερνετ, και κινητά τηλέφωνα, και εικονικά girlfriends.

Τόσοι ακριβώς. Ούτε ένας παραπάνω. Ούτε ένας παρακάτω.

Οι υπόλοιποι έκαναν και κάνουν αυτό που πάντα ήξεραν να κάνουν: Εχουν γνώμη για τα πάντα, τα ρίχνουν όλα στους ξένους, οσμίζονται πλεκτάνες, φαντασιώνονται μεγαλεία, βλέπουν παντού εχθρούς, νομίζουν πως όλοι τους χρωστάνε κάτι αλλά αυτοί δεν χρωστάνε τίποτα.

Κάποιοι από αυτούς θα σου μιλήσουν για τα λαμόγια τους πολιτικούς που μας κλέβουν, και μετά θα αποφύγουν να σου κόψουν απόδειξη.

Κάποιοι άλλοι θα σου μιλήσουν για το φιλόξενο πνεύμα του Ελληνα, και μετά θα πάνε να βάλουν φωτιά στα στρατόπεδα που θα μένουν οι πρόσφυγες.

Δεν έχει νόημα να ψάχνω για παραδείγματα. Ξέρετε ποιους λέω. Υπάρχουν παντού. Ζούνε ανάμεσά μας. Ζούνε ανάμεσά μας, δυστυχώς, και όχι σε παράλληλο σύμπαν όπως υποστήριζε σε πρόσφατο άρθρο του ο Κύριος Εκδότης.

Πολύ φοβάμαι πως η τεχνολογία (και οι θεοί) σηκώνουν τα χέρια ψηλά.

Αλλά το σημαντικό εδώ είναι το εξής.

Δεν είναι μόνο αυτοί που ζούνε ανάμεσά μας. Είμαστε κι εμείς που ζούμε ανάμεσά τους.

Είμαστε πλέον έτοιμοι. Εχουμε ετοιμάσει διάφορες σακούλες με παλιά ρούχα και σκεπάσματα, καθώς και σάντουιτς και κέικ σε αλουμινόχαρτο, σταφιδόψωμα, βραστά αυγά, και μπουκαλάκια με νερό. Ετοιμαζόμαστε να κατέβουμε από Πλατεία Βικτωρίας, στον καταυλισμό που έχει στηθεί, για να βοηθήσουμε όσο μπορούμε.

Αν έχετε χρόνο κάντε το κι εσείς.

Οι καιροί είναι δύσκολοι και καλό είναι να ξέρουμε πόσοι είμαστε.

Σας ασπάζομαι,
Θείος Ακάκιος


e-card
4000 1234 5678 9010
4000
GOOD THRU
MONTH / YEAR
12 / 19
MasterCard
Cardholder Name
Κάρτα e-coin, που δέχεται Bitcoins: Δουλεύει όπως όλες οι κανονικές κάρτες Debit, παντού στον κόσμο και για online αγορές.
Αποκτήστε τη δική σας με κλικ εδώ
και κερδίστε αυτομάτως
25% έκπτωση
κατά την παραγγελία της κάρτας.

Skill: Beginner
Tags: Indie games, game engine, crowdsourcing

Indie games και ανάπτυξη παιχνιδιών

Indie Games. Ή αλλιώς independent video games. Ή πιο σωστά, video games από independent developers. Κατά πάσα πιθανότητα έχετε ήδη παίξει κάποιο παιχνίδι που κατατάσσεται σε αυτή την κατηγορία – Minecraft ή Braid κανείς; Πόσοι από εσάς όμως έχετε ακούσει τον όρο;

του Γιάννη Κρομμύδα

Μιλώντας για indie game αναφερόμαστε σε παιχνίδια που έχουν γίνει είτε από ένα άτομο είτε από μικρές ομάδες ατόμων, χωρίς να έχουν βασιστεί σε χρηματοδότηση από εκδότριες εταιρείες. Στις μέρες μας, παιχνίδια τέτοιου τύπου γνωρίζουν άνθηση και σε αυτό βοηθούν αρκετοί παράγοντες. Καίρια συμβολή στην άνθηση αυτή έχει το Διαδίκτυο.



Δημιουργήθηκε από τον Σουηδό προγραμματιστή Markus "Notch" Persson και έκανε πάταγο στη διεθνή αγορά φτάνοντας –και ξεπερνώντας, θα λέγαμε– σε πωλήσεις παιχνίδια από μεγάλες εταιρείες παραγωγής παιχνιδιών.

Τα πιο ολοκληρωμένα πακέτα λογισμικού

Η δημιουργία ενός παιχνιδιού εμπλέκει δραστηριότητες από πολλές ειδικότητες. Από προγραμματιστές, γραφίστες και animators, μέχρι μουσικούς παραγωγούς, ειδικούς σε marketing και δικηγόρους. Όλοι μαζί και ο καθένας ξεχωριστά προσφέροντας τις υπηρεσίες τους, διαμορφώνουν την ποιότητα του τελικού προϊόντος. Όσοι σας λένε ότι είναι εύκολη δουλειά η δημιουργία ενός παιχνιδιού, ενδέχεται να αναφέρονται σε παιχνίδια μικρά σε περιεχόμενο και κατά συνέπεια σε μέγεθος.

Αρχικά, πρέπει να επισημανθεί ότι τα video games θεωρούνται από τα πιο ολοκληρωμένα πακέτα λογισμικού που μπορεί να βρει κάποιος στην αγορά. Συνδυάζουν σενάριο, γραφικά δύο και τριών διαστάσεων, user interface, animations και μουσική, όλα αυτά περιπλεγμένα με κώδικα έτσι ώστε όλα μαζί να δίνουν μια ξεχωριστή, ολοκληρωμένη εμπειρία στον παίκτη. Μπορεί η δημιουργία ενός ΑΑΑ (προφέρεται "triple A") τίτλου να μην είναι εύκολη υπόθεση, στις μέρες μας όμως είναι αρκετά εύκολο να δημιουργήσει κάποιος ένα μικρό παιχνίδι.

Ως ΑΑΑ χαρακτηρίζονται οι τίτλοι παιχνιδιών για τη δημιουργία των οποίων έχει δαπανηθεί ένας μεγάλος όγκος πόρων, όχι αποκλειστικά χρηματικών, και συνήθως είναι υψηλής ποιότητας. Τέτοια παιχνίδια τείνουν να λαμβάνουν υψηλές βαθμολογίες από τους κριτικούς του χώρου. Είναι σχεδόν αδύνατο να δημιουργηθούν από ένα μόνο άτομο, ενώ τα χρονοδιαγράμματά τους μπορούν να καλύψουν χρόνια. Άμεσο συμπέρασμα είναι ότι παιχνίδια σαν αυτά, κατά βάση, μπορούν να παραχθούν μόνο από μεγάλες εταιρείες, με μεγάλα κεφάλαια διαθέσιμα για την υλοποίησή τους.

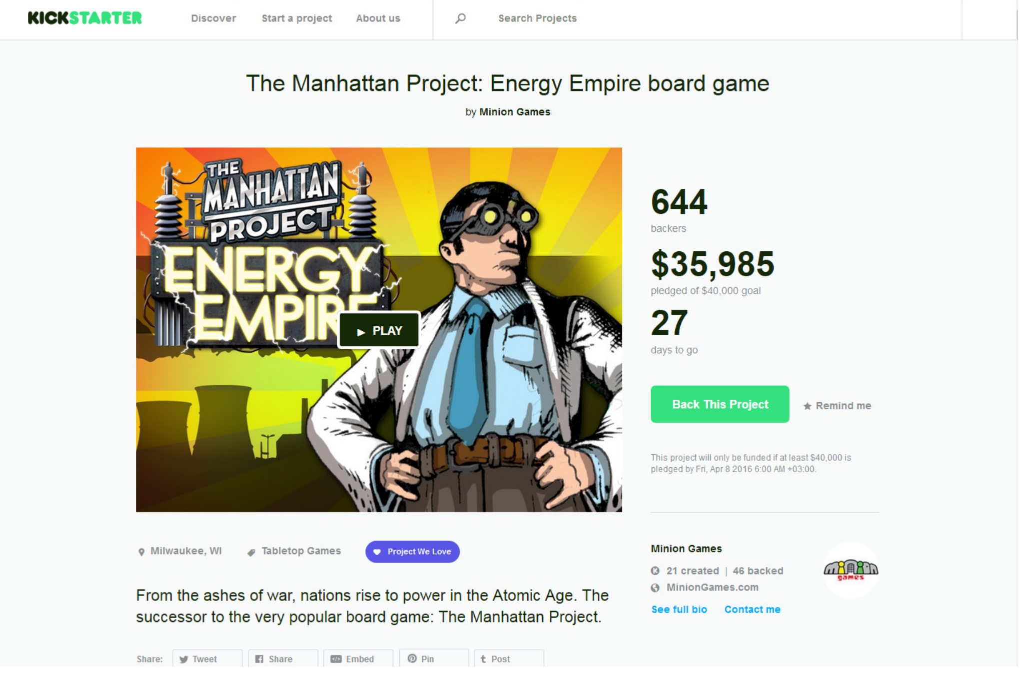
Και κάπου εδώ, λοιπόν, μπαίνουν στο παιχνίδι (pun intended) τα indie games.

Κατασκευή παιχνιδιών, χωρίς τους μεγάλους

Επειδή δεν γίνεται τα χρήματα –ή η έλλειψη αυτών– να καταπιέσουν το μεράκι, ομάδες ατόμων μιμούνται αυτές τις μεγάλες εταιρείες, παράγοντας παιχνίδια σε μικρότερη κλίμακα. Οι εν λόγω ομάδες δρουν χωρίς τη βοήθεια μεγάλων εταιρειών ή χωρίς να βασίζονται στις χρηματοδοτήσεις τους. Τους αποδίδεται η ονομασία independent, ελληνιστί "ανεξάρτητες", μιας και δεν υπάρχει καμία εξάρτηση από μεγαλύτερες εταιρείες. Δυστυχώς όμως, στην πορεία τους οι περισσότερες απ' αυτές τις ομάδες έρχονται αντιμέτωπες με προβλήματα που μεγάλες εταιρείες (ή εκδότες παιχνιδιών) έχουν λυμένα εξ' αρχής. Κυριότερα προβλήματα θεωρούνται το marketing και σαφώς η χρηματοδότηση.

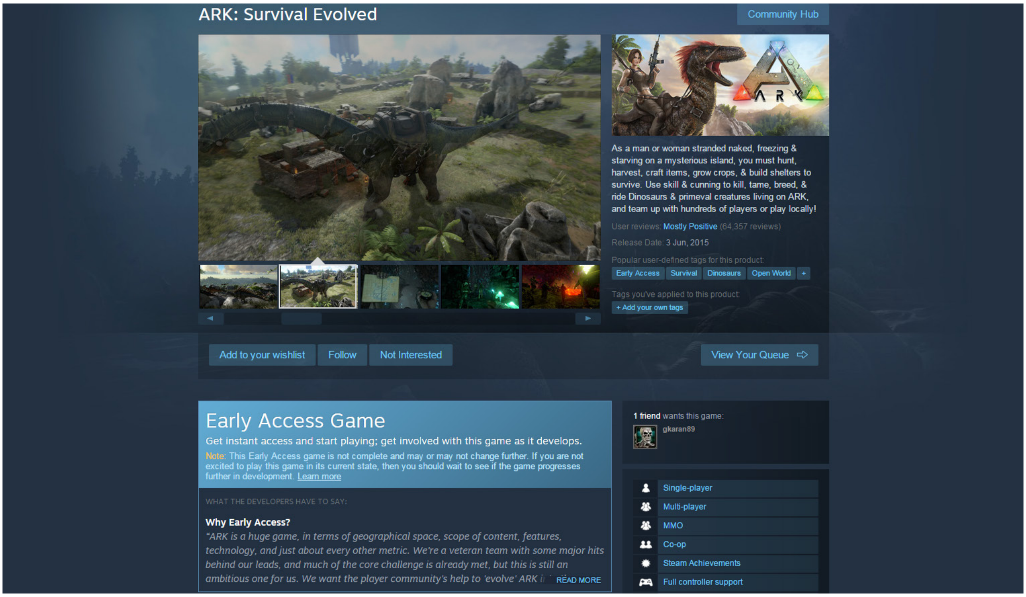
Τα πάντα σήμερα είναι marketing. Το marketing, η προώθηση δηλαδή, μπορεί να κρίνει τη μοίρα ενός παιχνιδιού. Ευτυχώς το Διαδίκτυο είναι εξαιρετικά διαδεδομένο Στις Μέρες Μας (TM) και οι διαφημίσεις σε αυτό δίνουν και παίρνουν. Σε σημείο που έχει καταντήσει να είναι πιο αποδοτική η διαφήμιση σ' αυτό, παρά όταν προβάλλεται στην τηλεόραση ή σε μια εφημερίδα. Είναι αρκετά εύκολο να διαφημιστεί κάποιος στο Internet, καθώς και να ρυθμίσει το μέγεθος της καμπάνιας του ανάλογα με τον οβολόν του.

Κάποιες ομάδες μπαίνουν στη διαδικασία να λύσουν το οικονομικό πρόβλημα της προσπάθειάς τους, κυνηγώντας χρηματοδότηση από το ευρύ κοινό. Ο αγγλικός όρος είναι crowdsourcing. Πρόκειται για μια μέθοδο χρηματοδότησης στην οποία μεγάλος αριθμός ατόμων διαθέτουν μικρά ποσά έκαστος για την υλοποίηση του έργου. Υπάρχουν ουκ ολίγες πλατφόρμες για το σκοπό αυτό, με μεγαλύτερες το Kickstarter και το Indiegogo. Σε αυτές οι developers μπορούν να παρουσιάσουν την ιδέα τους και να ζητήσουν από τους χρήστες να υποστηρίξουν την προσπάθειά τους, προαγοράζοντας ουσιαστικά το τελικό προϊόν.



Kickstarter: Το μεγαλύτερο και πιο γνωστό site crowdsourcing στο Διαδίκτυο. Φιλοξενεί πληθώρα από projects, με καινούργια να προστίθενται κάθε μέρα. Στην εικόνα βλέπουμε να έχει συγκεντρωθεί το ποσό των 36.000 (σχεδόν) δολαρίων. Σε αυτό συνέβαλαν 644 backers (υποστηρικτές), ενώ ο στόχος της εκστρατείας είναι να μαζευτούν τουλάχιστον 40.000 δολάρια. Απομένουν άλλες 27 ημέρες για τη λήξη της.

Άλλες φορές ζητούν τα χρήματα των παικτών εκ των προτέρων, βγάζοντας τα παιχνίδια τους σε Early Access. Ένα παιχνίδι σε Early Access είναι μεν playable, απλά δεν έχει να αποδώσει το 100% του περιεχομένου του ή της λειτουργικότητάς του. Για παράδειγμα, η Early Access έκδοση ενός FPS game, μπορεί να περιλαμβάνει μόνο μια πίστα και ένα μικρό αριθμό όπλων για να επιλέξει ο χρήστης. Παρόλο που ένα παιχνίδι στην κατάσταση αυτή μπορεί να παρουσιάζει έλλειψη περιεχομένου, η τιμή του εξακολουθεί να κυμαίνεται στα επίπεδα πώλησης ενός ολοκληρωμένου παιχνιδιού. Ο πρωταρχικός σκοπός του Early Access είναι να αποκτήσει η κατασκευάστρια ομάδα οικονομική ευχέρεια ώστε να είναι σε θέση να καλύψει κάποια από τα τρεχούμενα λειτουργικά της έξοδα.



Αν και ο όρος Early Access προέρχεται από την πλατφόρμα παιχνιδιών Steam της Valve, ουσιαστικά αναφέρεται σε όλα τα games που ενώ βρίσκονται ακόμα σε development στάδιο είναι διαθέσιμα για αγορά από το ευρύ κοινό.

Μία άλλη μέθοδος άξια αναφοράς που μάλιστα εμπίπτει και στις δύο προαναφερθείσες μεθόδους, είναι τα game bundles. Πολλά sites προσφέρουν "πακέτα" παιχνιδιών σε χαμηλές τιμές ή αφήνουν το χρήστη να ορίσει ο ίδιος πόσα χρήματα θέλει να δώσει. Βασίζονται στο ότι η χαμηλή τιμή θα προσελκύσει μεγαλύτερο αριθμό πωλήσεων και πολλές φορές το πετυχαίνουν. Ουκ ολίγες φορές όμως τα πακέτα αυτά περιλαμβάνουν μεγάλο αριθμό παιχνιδιών, με τους δημιουργούς τους να πρέπει να μοιραστούν το τελικό ποσό. Συνέπεια; Μηδαμινές αποδοχές σε σχέση με τον αριθμό των κομματιών που έχουν εν τέλει διατεθεί.

Ας δούμε τώρα καλύτερα πώς δημιουργείται ένα παιχνίδι.

Τα στάδια παραγωγής ενός παιχνιδιού

Η δημιουργία ενός παιχνιδιού ξεκινάει από τη φάση της σχεδίασης. Σε αυτή τη φάση οι game designers συγγράφουν το game design document, με τους concept artists να δουλεύουν σε στενή συνεργασία μαζί τους. Η δουλειά των τελευταίων είναι να χρησιμοποιήσουν την φαντασία τους, μεταφέροντας στο χαρτί τις σκέψεις των designers. Με τη διαδικασία αυτή το παιχνίδι αρχίζει να λαμβάνει μια πρώτη μορφή από άποψη περιεχομένου. Πολλές φορές στη φάση αυτή δημιουργούνται και κάποια prototypes του παιχνιδιού.

Τα game design documents είναι έγγραφα που περιλαμβάνουν, με εξουθενωτικές λεπτομέρειες, πληροφορίες για το παιχνίδι καθώς και τις λειτουργίες του. Περιλαμβάνουν από τις πιο γενικές έννοιες που μπορούν να υπάρχουν στο παιχνίδι (π.χ.,

την όλη ατμόσφαιρα που θα κυριαρχεί σε αυτό), μέχρι και την παραμικρή λεπτομέρεια στο gameplay (π.χ., τον τρόπο υπολογισμού της ζημιάς που θα πάθει μια μονάδα αν της επιτεθεί μια άλλη).

Prototypes ή game prototypes, ονομάζονται οι εκδόσεις των παιχνιδιών που σαν στόχο έχουν να τεστάρουν τη βιωσιμότητα μιας ιδέας. Συνήθως οι εκδόσεις αυτές χρησιμοποιούν "quick and dirty code", λόγω των στενών χρονικών περιθωρίων που επιβάλλονται για την ολοκλήρωσή τους. Βασική τους δουλειά είναι να βοηθήσουν στην περαιτέρω οπτικοποίηση της ιδέας, υλοποιώντας τους βασικούς μηχανισμούς του παιχνιδιού.

Όταν η φάση της σχεδίασης φτάσει σε ένα ώριμο στάδιο, ξεκινάει η διαδικασία παραγωγής. Σε αυτή τη φάση δημιουργούνται ο κώδικας, καθώς και τα υπόλοιπα assets του παιχνιδιού (γραφικά, μουσική, animations, διάλογοι κ.λπ.). Η φάση αυτή απαιτεί το μεγαλύτερο ποσοστό του χρόνου της διαδικασίας παραγωγής.

Αμέσως μετά ξεκινάει η διαδικασία της δοκιμής του παιχνιδιού, μέχρι να ολοκληρωθεί στο 100% το περιεχόμενό του. Testing μέχρι τελικής πτώσεως και το παιχνίδι είναι έτοιμο προς release.

Ας σταθούμε περισσότερο στη δεύτερη φάση της διαδικασίας αυτής, μιας και μας αφορά περισσότερο (όντας προγραμματιστές).

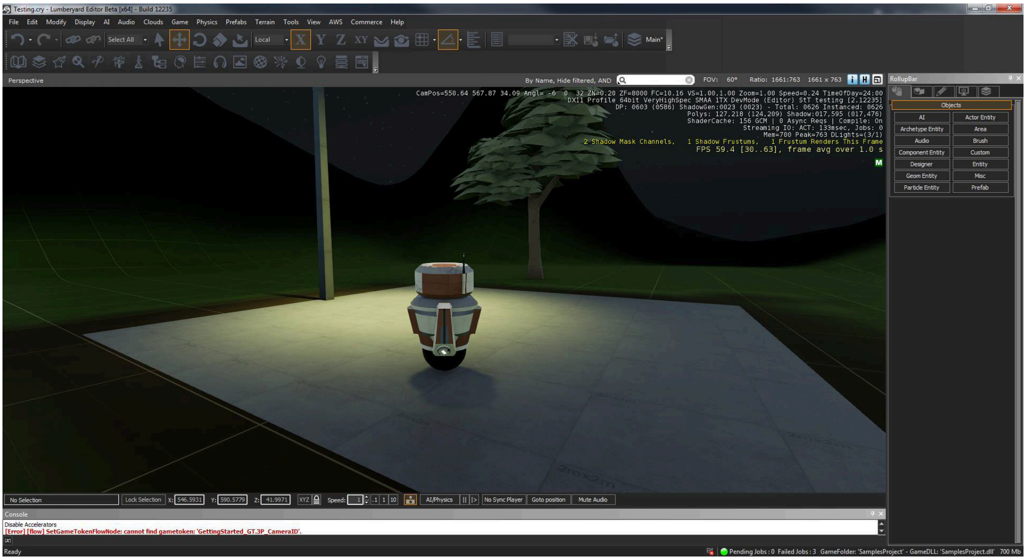
Η πολύτιμη game engine

Πίσω από κάθε παιχνίδι κρύβεται μια μηχανή. Πρόκειται για τη λεγόμενη game engine, η οποία είναι υπεύθυνη για οτιδήποτε συμβαίνει στο σύστημα που τρέχει το εν λόγω παιχνίδι. Είναι μια συλλογή άλλων μηχανών (graphics engine, audio engine κ.ά.), οι οποίες λειτουργούν μεταξύ τους σε πλήρη εναρμόνιση. Μια game engine προσφέρει λειτουργίες που μπορεί να περιλαμβάνουν:

- το rendering των γραφικών στην οθόνη (graphics)
- την αναπαραγωγή της μουσικής και ηχητικών εφέ (audio)
- την επικοινωνία πολλών υπολογιστών μεταξύ τους (networking)
- το πόση ενέργεια θα χάσει ο αντίπαλος στρατιώτης όταν θα τον χτυπήσει η σφαίρα που έφυγε από το δικό μας όπλο, όταν κάναμε κλικ με το ποντίκι μας (input)

Αυτό το τελευταίο είναι ευρύτερα γνωστό με τον όρο gameplay.

Η κατασκευή μιας τέτοιας μηχανής είναι εξαιρετικά χρονοβόρα και επίπονη διαδικασία, οπότε σαν προγραμματιστές που σεβόμαστε τον χρόνο μας καλό είναι να στρεφόμαστε σε πιο άμεσες λύσεις. Σε όσα θα ακολουθήσουν, θα χρησιμοποιήσουμε μια έτοιμη game engine.



Lumberyard: Η νέα μηχανή δημιουργίας παιχνιδιών της Amazon. Είναι Open source – και πολλά υποσχόμενη!

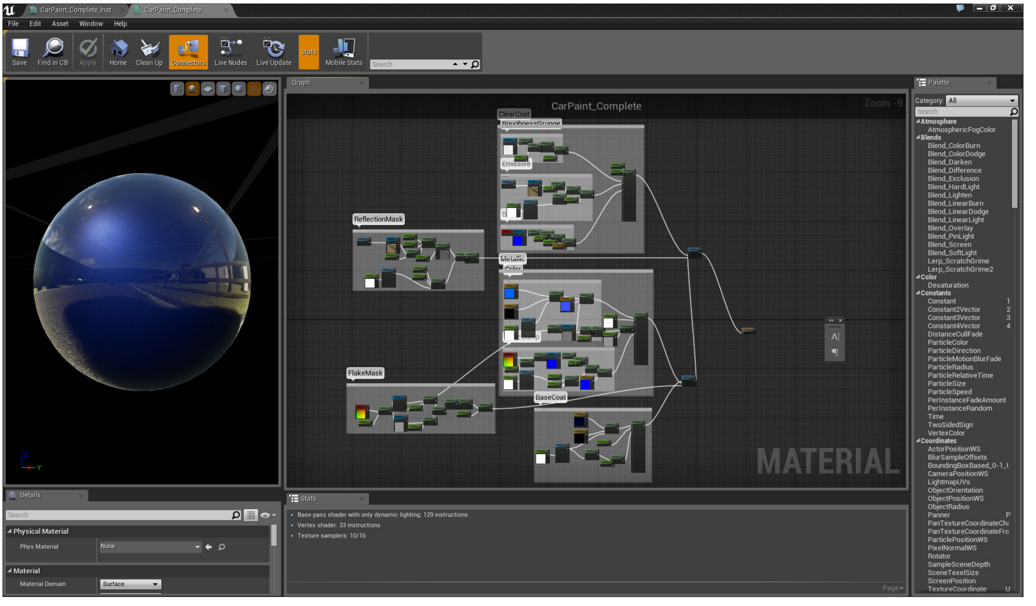
Στο Διαδίκτυο κυκλοφορούν πολλές τέτοιες μηχανές και μάλιστα αρκετές από αυτές διατίθενται δωρεάν. Εφόσον η κάθε μία τους αποτελεί διαφορετικό λογισμικό, διαθέτουν διαφορετικές δυνατότητες και παρουσιάζουν διαφορετικές καμπύλες εκμάθησης. Συχνά έχουν διαφορετικές απαιτήσεις σε προγραμματιστικές γνώσεις, ενώ υποστηρίζουν και διαφορετικές γλώσσες προγραμματισμού. Πολλές εξάγουν τα παιχνίδια που δημιουργούνται με αυτές σε περισσότερες από μία πλατφόρμες (PC, Mac, Xbox, PS4, Android κ.ά.), απαιτώντας μόνο μικρές τροποποιήσεις ή προσθήκες στον κώδικα.

Πολλοί κατασκευαστές game engines δίνουν το πρόγραμμα της μηχανής δωρεάν προς χρήση, με την προϋπόθεση να εισπράξουν ποσοστό των χρημάτων από τις πωλήσεις του παιχνιδιού. Ακούγεται άσχημο; Ίσως και να είναι, αλλά συμβαίνει μόνο σε περίπτωση που τα έσοδα των πωλήσεων ξεπεράσουν ένα ορισμένο από την εταιρεία ποσό (π.χ., 100.000 δολάρια).

Πέρα από τη μηχανή, οι προγραμματιστές πρέπει να γράψουν και τον κώδικα των λειτουργιών του παιχνιδιού (gameplay). Σε σύγκριση με τον κώδικα της μηχανής, ο κώδικας του gameplay είναι πιο εύκολος. Ουσιαστικά, χρησιμοποιεί το API της μηχανής για να συνδέσει την είσοδο του παίχτη (ποντίκι, πληκτρολόγιο, επιταχυνσιόμετρο κ.λπ.) με τη συμπεριφορά και την αλληλεπίδραση των assets που εμφανίζονται στην οθόνη όταν εκτελείται το παιχνίδι.

Στα επόμενα άρθρα θα δημιουργήσουμε από την αρχή ένα μικρό παιχνίδι. Θα στοχεύσουμε κυρίως σε mobile συσκευές.

Για να τρέχει το παιχνίδι μας στα κυριότερα λειτουργικά συστήματα των κινητών συσκευών με *ένα* codebase, θα το προγραμματίσουμε σε μια μηχανή που ακούει στο όνομα Unity. Υποστηρίζει τη δημιουργία scripts σε Boo (μια διάλεκτο της Python), σε



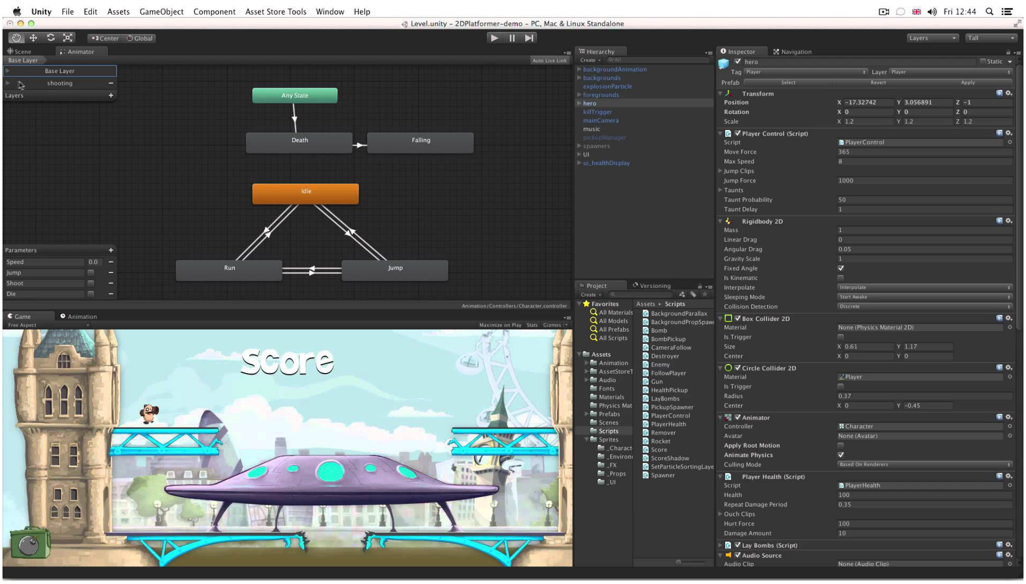
Η Unreal Engine της Epic Games. Η μηχανή σχεδιάστηκε αρχικά για FPS games και έχει πολλούς και καλούς τίτλους στην πλάτη της. Πήρε το όνομά της από το παιχνίδι Unreal (γύρω στο '98, όσοι το θυμούνται).

UnityScript (θυμίζει πολύ Javascript αλλά χρησιμοποιεί classes) και σε C#. Εμείς θα χρησιμοποιήσουμε την τελευταία, ενώ προτείνεται η χρήση του Visual Studio (κατά την εγκατάσταση της μηχανής, παρέχεται δυνατότητα εγκατάστασής του). Αν δεν θέλει κάποιος να κατεβάσει τα αρκετά gigabytes του Visual Studio, μαζί με τη μηχανή εγκαθίσταται και το MonoDevelop, ένα εναλλακτικό IDE για συγγραφή κώδικα. Η μηχανή χρησιμοποιεί ακόμα την έκδοση 2 του Mono Project (έκδοση 3.5 του .NET framework – δυστυχώς), ενώ σε κάποιες πλατφόρμες λειτουργεί με Subset της έκδοσης αυτής.

Το documentation της μηχανής είναι εκτενέστατο. Είναι διαθέσιμο online, αλλά μας παρέχεται δυνατότητα να το αποκτήσουμε και offline κατά την εγκατάστασή της. Για να δημιουργήσουμε τα τελικά εκτελέσιμα αρχεία του παιχνιδιού χρειάζονται τα αντίστοιχα SDKs του λειτουργικού συστήματος που έχουμε ως στόχο.

Για παράδειγμα, δεν γίνεται να κάνουμε build για iOS σε μηχάνημα διαφορετικό από Mac. Το Hackintosh δεν θα δουλέψει στην συγκεκριμένη περίπτωση, γιατί το Xcode δεν είναι είναι διαθέσιμο (και δεν μπορεί να εγκατασταθεί) σε αυτό. Αν θέλουμε να κάνουμε build σε Windows Phone, θα χρειαστεί οπωσδήποτε το Windows Phone 8.1 SDK ή το Windows 10 SDK, ανάλογα με την έκδοση και το Visual Studio. Το δε build για Android γίνεται από παντού, απλά θα χρειαστεί το SDK του.

Το testing του παιχνιδιού θα το κάνουμε στον ίδιο τον υπολογιστή μας, οπότε δεν υπάρχει λόγος ανησυχίας αν δεν έχουμε πρόσβαση ή αν δεν θελήσουμε να κατεβάσουμε όλα τα extra που χρειάζονται για τα κανονικά builds.



Η μηχανή είναι αρκετά χρόνια στο προσκήνιο, βρίσκεται στην 5η έκδοσή της, ενώ οι χρήστες της πολλαπλασιάζονται μέρα με τη μέρα. Υπάρχουν εκδόσεις της για Windows και OS X, ενώ στα σκαριά βρίσκεται μια έκδοση που θα τρέχει σε Linux.

Τι ακολουθεί

Στα άρθρα μας θα εμβαθύνουμε περισσότερο στον προγραμματισμό και στις τεχνικές που αφορούν στη συγκεκριμένη μηχανή. Φιλική προειδοποίηση: το παιχνίδι θα περιέχει programmer art. Με τον όρο "programmer art" αναφερόμαστε σε γραφικά που χρησιμοποιούνται κατά τη διάρκεια της παραγωγής του κώδικα και είναι φτιαγμένα από τους ίδιους τους προγραμματιστές. Για παράδειγμα, τον παίχτη μπορεί να παριστάνει μια 3D κάψουλα χρώματος μπλε, ενώ ένα σπίτι μπορεί να είναι ένας απλός κύβος, χρώματος γκρι. Αν χρειαστούν ιδιαίτερα γραφικά θα καταφύγουμε σε πακέτα με δωρεάν assets, πληθώρα εκ των οποίων υπάρχουν στο Asset Store της Unity.

Τα assets του project θα είναι διαθέσιμα ανά πάσα στιγμή σε σελίδα στο GitHub (pull requests welcome). Για τη συνέχεια, μεταβείτε στο άρθρο που αρχίζει από τη σελίδα 38 του τεύχους.

Skill: Intermediate
Tags: Sampling frequency, Direct Digital Synthesis, tempo, Beats per Minute, phase-delta

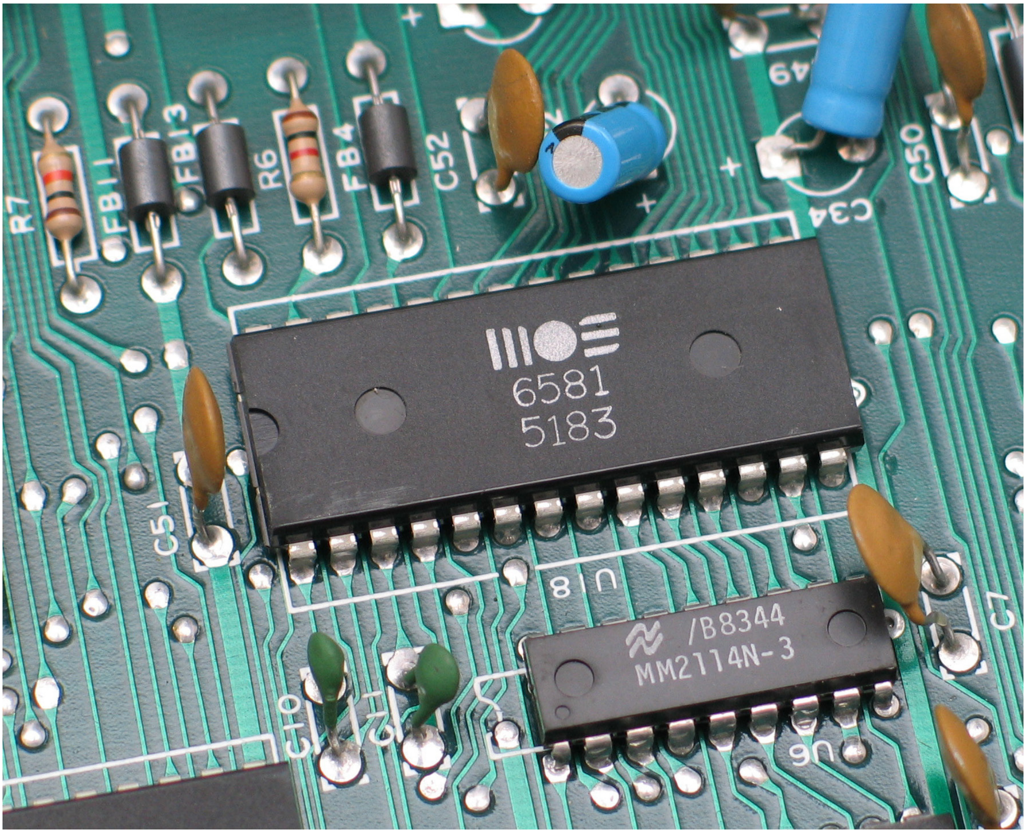
Κατασκευάστε έναν υπολογιστή, μέρος 5:

Ηλεκτρικά γρατζουνίσματα

Το μικρό PC που βαλθήκαμε να κατασκευάσουμε με μερικούς μόνο μικροελεγκτές, διαθέτει και κάρτα ήχου. Μην περιμένετε ν' ακούσετε ήχο υψηλής πιστότητας, αλλά να είστε σίγουροι ότι αν ασχοληθείτε μ' αυτή θα κολλήσετε. Το firmware της κάρτας μπορεί να συνθέσει ηλεκτρονικές μελωδίες, σαν αυτές που συνόδευαν τα παλιά Arcade games.

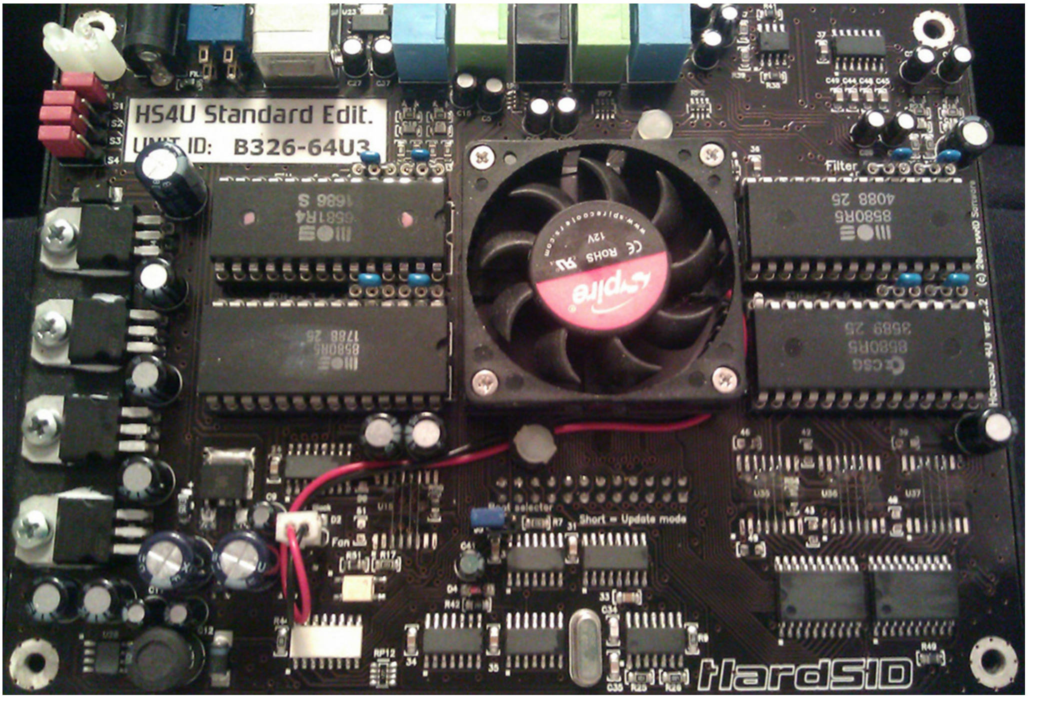
του Παναγιώτη Βαρελά

Τα πρώτα PC ενσωμάτωναν μόνο ένα μικρό ηχείο, από το οποίο ακούγονταν μονότονα, βαρετά μπιπ. Εμείς βέβαια, κατασκευάζοντας τον δικό μας υπολογιστή, δεν έχουμε κανένα λόγο να περιοριστούμε στον κόσμο των PC. Την ίδια εποχή κυκλοφορούσαν και άλλοι υπολογιστές, που διέθεταν εξειδικευμένα κυκλώματα ήχου και παιχνίδια με εντυπωσιακή μουσική. Προφανώς, εκείνα τα τσιπάκια ήχου προσέφεραν στοιχειώδεις λειτουργίες και δεν θύμιζαν σε τίποτα τους σύγχρονους απογόνους τους. Σκεφτείτε ότι τα περισσότερα μπορούσαν να παράγουν μόνο τετραγωνικά ή τριγωνικά κύματα, ενώ σ' ορισμένες περιπτώσεις προσέφεραν και πέντε-έξι υποτυπώδη εφέ. Οι προγραμματιστές της εποχής, για να συνθέσουν τις μελωδίες που συνόδευαν τα παιχνίδια συνδύαζαν τους ελάχιστους διαθέσιμους πόρους με ευφάνταστους τρόπους. Εξάλλου, πολύ συχνά κατέληγαν να χακάρουν τα τσιπ, αφού τα χρησιμοποιούσαν με τρόπους δεν είχαν προβλέψει ούτε οι κατασκευαστές τους. Κάτι τέτοιο συνέβη και με το κύκλωμα ήχου του Commodore 64, το περίφημο SID (Sound Interface Device) 6581/8580. Η τεράστια εμπορική επιτυχία του εν λόγω υπολογιστή και τα αμέτρητα παιχνίδια που κυκλοφόρησαν γι' αυτόν, έκαναν το συγκεκριμένο κύκλωμα ιδιαίτερα δημοφιλές. Αρκεί να σας πούμε ότι υπάρχουν προγραμματιστές/μουσικοί που συνθέτουν κομμάτια για το συγκεκριμένο τσιπάκι μέχρι σήμερα. Τα περίφημα chiptunes και η σχετική σκηνή (demo scene) επηρεάστηκαν σε μεγάλο βαθμό από την ύπαρξη του SID.



Το τσιπάκι ήχου SID, επάνω στη μητρική του Commodore 64. Πρόκειται για έναν υπολογιστή που σχεδόν σαράντα χρόνια μετά διατηρεί παγκόσμιο ρεκόρ πωλήσεων.

Το δικό μας κύκλωμα ήχου δεν διαφέρει πολύ από εκείνα που έδιναν φωνή στους πρώτους "οικιακούς υπολογιστές". Όπως υποψιάζεστε, και για εμάς πηγή έμπνευσης αποτέλεσε το SID. Η εν λόγω κάρτα –αν και θα έπρεπε να μιλάμε για το firmware που τρέχει στον μικροελεγκτή–, προσφέρει τέσσερα ανεξάρτητα κανάλια ήχου, που μπορούν να παίζουν ταυτόχρονα. Το ένα από αυτά παράγει μόνο τριγωνικά κύματα, τα άλλα δύο περιορίζονται σε τετραγωνικά κύματα, ενώ το τέταρτο μπορεί να παράγει μόνο θόρυβο. Επιπρόσθετα, προσφέρονται τα εφέ pitch-bend up, pitch-bend down και vibrato. Το κύκλωμα ήχου μπορεί να παίζει όποιες νότες θέλουμε, σε ένα ή και σε περισσότερα κανάλια ταυτόχρονα, εφαρμόζοντας όποιο εφέ επιθυμούμε σε κάθε νότα ξεχωριστά. Πρόκειται για ένα εξαιρετικά φτωχό αντίγραφο του SID, αλλά αν έχετε όρεξη και την απαιτούμενη έμπνευση, να είστε σίγουροι ότι θα μεγαλουργήσετε. Τι, δεν μας πιστεύετε; Αν όχι κατεβάστε το archive από το https://deltahacker.gr/?p=15994 κι ακούστε τι έγραψε ένας φίλος μουσικός, "επάνω" στην κάρτα ήχου του υπολογιστή μας.



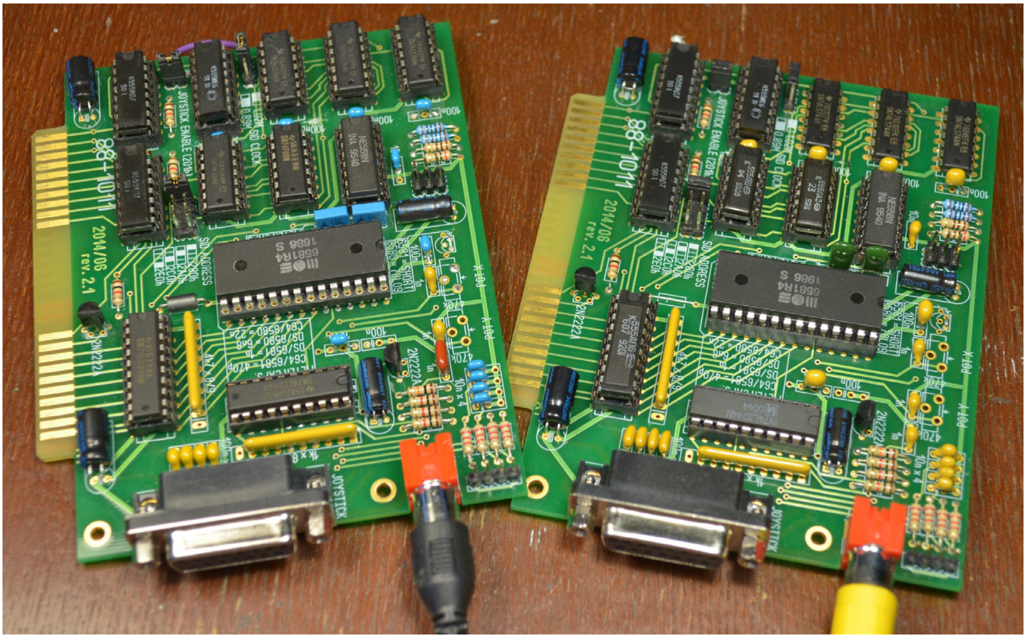
Το SID χρησιμοποιείται ακόμα και σήμερα, από τους νοσταλγούς της παλιάς ηλεκτρονικής μουσικής. Στη φωτογραφία φαίνεται το εσωτερικό του HardSID Quatro. Πρόκειται για μια εξωτερική κάρτα ήχου που στελεχώνεται από τέσσερα SID(!) και συνεργάζεται με τα περισσότερα software synthesizers που υποστηρίζουν το SID. Ναι, υπάρχουν πολλά soft-synth που υποστηρίζουν το SID και όταν δεν υπάρχει διαθέσιμο, το εξομοιώνουν.

Αντίστροφη πορεία

Όπως είδαμε στο ξεκίνημα της σειράς, το κύκλωμα ήχου δεν περιλαμβάνει κανένα εξωτικό ολοκληρωμένο. Απαρτίζεται από έναν ATmega328 (ο μικροελεγκτής του Arduino) και ένα δίκτυο αντιστάσεων, που λειτουργεί σαν υποτυπώδης DAC (Digital to Analog Converter). Μπορείτε να διαβάσετε περισσότερα για τη λειτουργία αυτού του μετατροπέα στο άρθρο "Από τη θεωρία στην πλακέτα", του τεύχους 047.

Η σύνθεση του ήχου, λοιπόν, πραγματοποιείται από το πρόγραμμα που τρέχει στον μικροελεγκτή και η τεχνική που επιλέξαμε γι' αυτή τη δουλειά ονομάζεται DDS (Direct Digital Synthesis). Για να κατανοήσετε εύκολα την κεντρική ιδέα πίσω από αυτή τη μέθοδο, σκεφτείτε για λίγο την αντίστροφη διαδικασία: την ψηφιοποίηση του ήχου. Αρχικά, με τη βοήθεια ενός μικροφώνου, οι μηχανικές ταλαντώσεις που συνιστούν τον ήχο μετατρέπονται σε ηλεκτρικές ταλαντώσεις. Στη συνέχεια, αυτό το ηλεκτρικό σήμα οδηγείται σ' έναν ADC (Analog to Digital Converter). Ο συγκεκριμένος μετατροπέας λαμβάνει αλλεπάλληλα δείγματα από το εισερχόμενο σήμα, με μια συχνότητα που ονομάζεται "συχνότητα δειγματοληψίας" (sampling frequency). Παρεμπιπτόντως, μια τυπική τιμή αυτού του μεγέθους είναι τα 44100Hz – όλο και κάπου θα την έχει πάρει το μάτι σας. Για κάθε δείγμα που λαμβάνει ο ADC παράγεται ένας αριθμός που περιγράφει τη στιγμιαία τιμή της τάσης. Οι αριθμοί

που προκύπτουν κατ' αυτόν τον τρόπο αποτελούν την ψηφιακή αναπαράσταση της ηλεκτρικής ταλάντωσης και, κατ' επέκταση, περιγράφουν το αρχικό ηχητικό κύμα. Πίσω από την προηγούμενη περιγραφή κρύβονται πολλές σημαντικές λεπτομέρειες, αλλά δεν υπάρχει λόγος να επεκταθούμε. Αρκεί να έχετε υπόψη τη γενική ιδέα. Η μέθοδος DDS προβλέπει την ακριβώς αντίστροφη εργασία. Για να συνθέσουμε έναν ήχο υπολογίζουμε όλες εκείνες τις αριθμητικές τιμές, τις οποίες αν μετατρέψουμε σε τάσεις θα πάρουμε μια ηλεκτρική ταλάντωση όμοια με το ζητούμενο ηχητικό κύμα. Για παράδειγμα, αν θέλουμε να συνθέσουμε τη νότα που ονομάζεται "μεσαία Λα", πρέπει να συνθέσουμε ένα αρμονικό κύμα συχνότητας 440Hz. Επομένως, πρέπει να ξεκινήσουμε υπολογίζοντας τις διαδοχικές τιμές μιας ημιτονοειδούς συνάρτησης. Στη συνέχεια πρέπει να μετατρέψουμε αυτές τις τιμές σε τάσεις, με κατάλληλο ρυθμό, ώστε να προκύψει μια ηλεκτρική ταλάντωση συχνότητας 440Hz.



Το 1989 κυκλοφόρησε μια κάρτα ήχου για τον δίαυλο ISA, που στηριζόταν στο SID. Παρεμπιπτόντως, ο δίαυλος ISA είναι πρόγονος του διαύλου PCI, ο οποίος με τη σειρά του είναι πρόγονος του PCI Express. Στη φωτογραφία βλέπουμε δύο κλώνους της (πανάρχαιας) κάρτας ήχου, κατασκευασμένους το 2010 από λάτρεις του SID.

Υλικά μιας άλλης εποχής

Περιγράφοντας τα πρώτα κυκλώματα ήχου, όπως και τις δυνατότητες του δικού μας, ενδέχεται να προέκυψαν ορισμένες απορίες. Τι σχέση έχουν τα τριγωνικά και τα τετραγωνικά κύματα με τον ήχο; Τι δουλειά έχει ο θόρυβος σε μια κάρτα ήχου;

Θεωρητικά, ο ήχος μιας νότας αποτελεί ένα απλό αρμονικό κύμα. Ακριβώς γι' αυτό, συνηθίζουμε να λέμε ότι πρόκειται για ένα ημιτονοειδές κύμα και το περιγράφουμε μαθηματικά με τη βοήθεια κάποιας τριγωνομετρικής συνάρτησης. Ας επιστρέψουμε όμως στο θέμα μας. Οι παλιοί υπολογιστές δεν προλάβαιναν

να υπολογίσουν τις διαδοχικές τιμές μιας ημιτονοειδούς συνάρτησης, ώστε να συνθέσουν το αντίστοιχο κύμα σε πραγματικό χρόνο. Έτσι, οι σχεδιαστές των κυκλωμάτων ήχου είχαν καταφύγει σε μια πονηρή λύση: Αντί να παράγουν ημιτονοειδή κύματα, παρήγαγαν τριγωνικά (triangle waves). Ο ήχος των τριγωνικών κυμάτων μοιάζει αρκετά με τον ήχο των αρμονικών, αλλά δεν είναι καθαρός. Το μεγάλο του προτέρημα είναι ότι παράγεται πανεύκολα. Σκεφτείτε ότι όλη η δουλειά του hardware περιορίζεται σε διαδοχικές προσθέσεις και αφαιρέσεις, που όπως και να το κάνουμε είναι πολύ ευκολότερες από τον υπολογισμό των τιμών μιας τριγωνομετρικής συνάρτησης. Αυτή η έκπτωση στην ποιότητα του ήχου ήταν απόλυτα αποδεκτή για την εποχή, αλλά δεν αρκούσε για να καλύψει τις σχετικές ανάγκες. Εκείνοι που συνέθεταν μουσική για τα παιχνίδια χρειάζονταν αρκετά διαφορετικά "μουσικά όργανα". Οι σχεδιαστές του hardware βρήκαν πάλι τη λύση: τετραγωνικά κύματα (square waves). Όπως αντιλαμβάνεστε, ο ήχος αυτών των κυμάτων είναι ακόμα πιο τραχύς από τον ήχο των τριγωνικών, αλλά η χρήση τους κρύβει δυο σημαντικά πλεονεκτήματα. Αφενός, ένα τετραγωνικό κύμα παράγεται πολύ πιο εύκολα από ένα τριγωνικό, αφού σχηματίζεται από δύο μόνο τιμές. Αφετέρου, μεταβάλλοντας το duty cycle (δηλαδή τη χρονική διάρκεια της μίας στάθμης σε σχέση με την περίοδο του κύματος), η χροιά του ήχου αλλοιώνεται σε μεγάλο βαθμό και μοιάζει να προέρχεται από διαφορετικό όργανο. Ως εκ τούτου, τα τσιπάκια που παρήγαγαν τετραγωνικά κύματα προσέφεραν τον ήχο διαφορετικών οργάνων, αξιοποιώντας το ίδιο ακριβώς υποσύστημα. Για το τέλος αφήσαμε το θόρυβο. Οι μουσικοί τον χρησιμοποιούσαν για να μιμηθούν τον ήχο των τυμπάνων (με αρκετή φαντασία, ομολογουμένως) καθώς και για την ηχητική επένδυση ειδικών καταστάσεων/συμβάντων στα παιχνίδια. Γι' αυτό το λόγο, πέρα από τον εγγενή θόρυβο που υπήρχε ούτως ή άλλως στα κυκλώματα ήχου, οι μηχανικοί ενσωμάτωναν και γεννήτριες θορύβου.

Ζήτημα ποιότητας

Έχετε δει ποτέ τη σύγκριση ψηφιακών φωτογραφικών μηχανών, με βάση τα Megapixels; Περιττό να πούμε ότι μια τέτοια σύγκριση είναι πέρα ως πέρα λανθασμένη, αφού η τελική ποιότητα της εικόνας καθορίζεται κατά κύριο λόγο από το φακό. Τι να τα κάνεις τα 10 Gigapixels, αν η εικόνα που φτάνει στον αισθητήρα είναι θολή; Δεν προσπαθούμε να σας πείσουμε ότι το πλήθος των pixels είναι αδιάφορο. Η ανάλυση του αισθητήρα καθορίζει το άνω όριο στην ποιότητα των φωτογραφιών. Όμως, η ποιότητα καθορίζεται περισσότερο από τη δυνατότητα σωστής εστίασης κι από την ποσότητα του φωτός που φτάνει στον αισθητήρα. Με άλλα λόγια, τα πολλά Megapixels δεν δίνουν πάντα καλύτερες φωτογραφίες. Αναρωτιέστε γιατί αλλάξαμε θέμα; Είναι που σχεδιάζοντας τον synthesizer ο οποίος δίνει φωνή στην κάρτα ήχου, κινδυνέψαμε να πέσουμε στην ίδια παγίδα.

Η κεντρική ιδέα πίσω από τη μέθοδο DDS είναι αρκετά απλή. Όμως στο παράδειγμα που αναφέραμε υπήρχε μια ασάφεια. Τι σημαίνει η φράση "με κατάλληλο ρυθμό"; Στη διαδικασία της ψηφιοποίησης, ο ρυθμός δειγματοληψίας είναι σταθερός και η επιλογή του γίνεται σύμφωνα με το θεώρημα της δειγματοληψίας (γνωστό και ως "Θεώρημα Whittaker – Nyquist – Kotelnikov – Shannon"). Το εν λόγω θεώρημα καθορίζει μια ελάχιστη τιμή για τη συχνότητα δειγματοληψίας κι αν προσπεράσουμε

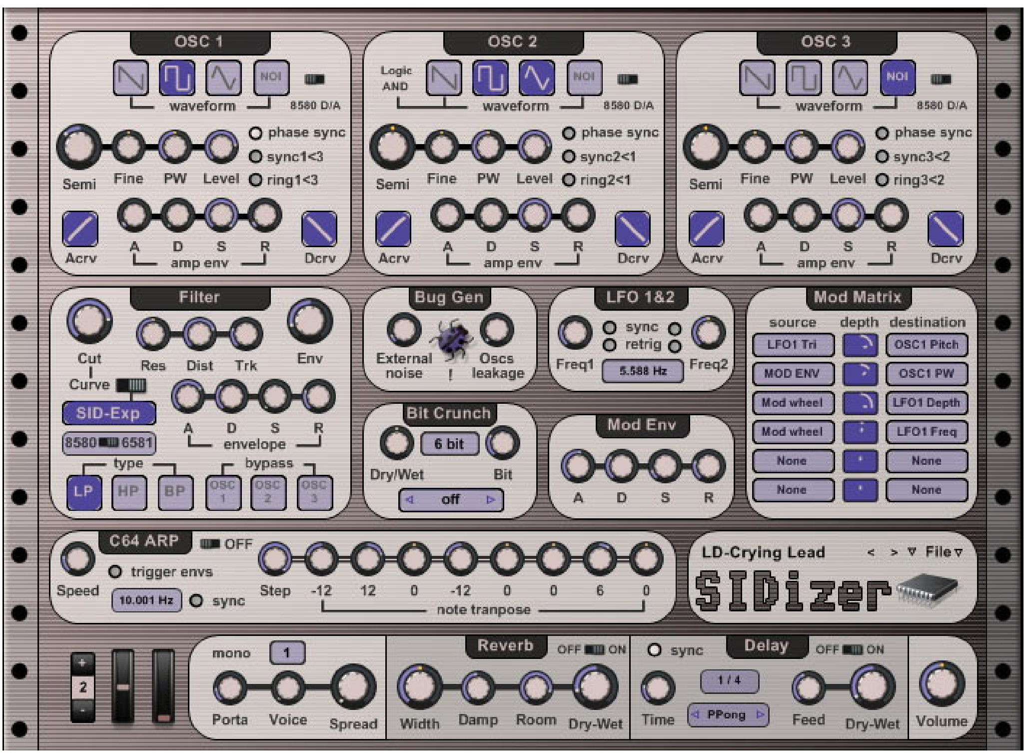


Αρχικά συνδέσαμε την έξοδο της κάρτας ήχου του μικρού υπολογιστή, στην είσοδο της κάρτας ήχου του μεγάλου. Μετά βάλαμε τον μικρό υπολογιστή να παίξει το theme του Bubble Bobble και τον μεγάλο υπολογιστή να αποθηκεύσει το εισερχόμενο σήμα. Στην εικόνα φαίνεται ένα στιγμιότυπο της κυματομορφής. Όταν ακούς και τη μουσική, τα χοντροκομμένα τετράγωνα γίνονται ξαφνικά πολύ πιο όμορφα!

τις μαθηματικές διατυπώσεις, η κεντρική ιδέα έχει ως εξής: Όσο πιο "πυκνά" είναι τα δείγματα που λαμβάνουμε, τόσο πιο πιστή θα είναι και η περιγραφή του ηχητικού κύματος. (Σ.τ.Ε. Είδατε τα μαθηματικά, που πάντα βγάζουν νόημα;) Αυτό μπορεί να το αντιληφθεί κανείς και διαισθητικά. Η συχνότητα δειγματοληψίας, λοιπόν, είναι σταθερή κι όσο μεγαλύτερη τιμή επιλέγουμε, τόσο ποιοτικότερη είναι και η αναπαράσταση του ήχου. Καλά όλα αυτά, θα πει κανείς, αλλά εμείς εξετάζουμε τη σύνθεση και όχι την ψηφιοποίηση. Όταν αναπαράγουμε ψηφιακό ήχο, τα δείγματα μετατρέπονται σε τάσεις με την ίδια συχνότητα που τα είχαμε λάβει. Έτσι δεν είναι; Εμμέσως, το θεώρημα της δειγματοληψίας εφαρμόζεται και στην αναπαραγωγή. Αν το σκεφτείτε λίγο, θα διαπιστώσετε ότι η αναπαραγωγή δεν διαφέρει ιδιαίτερα από τη σύνθεση. Στη δεύτερη περίπτωση τα δείγματα είναι κατασκευασμένα, αλλά αυτό δεν καταργεί την ανάγκη για μεγάλο και "πυκνό" πλήθος. Τελικά, φαίνεται ότι το θεώρημα της δειγματοληψίας επηρεάζει και τη σύνθεση.

Μήπως τώρα εντοπίζετε την αναλογία με τις ψηφιακές φωτογραφικές; Το πλήθος των δειγμάτων που παράγει ένα synthesizer ανά μονάδα χρόνου, επηρεάζει την ποιότητα του παραγόμενου ήχου. Ωστόσο, αν αυτά τα δείγματα έχουν κατασκευαστεί με λανθασμένο τρόπο, τότε όσα κι αν είναι το τελικό αποτέλεσμα θα περιέχει ατέλειες. Ο ρυθμός με τον οποίο παράγονται τα δείγματα καθορίζει το άνω όριο

στην ποιότητα του ήχου, αλλά δεν μπορεί να το επιβάλλει. Εν ολίγοις, δεν υπάρχει κανένας λόγος να κυνηγάμε τις μεγάλες συχνότητες. Ειδικά στην περίπτωσή μας, τα δείγματα του ήχου παράγονται από τον μικροελεγκτή σε πραγματικό χρόνο κι αυτό δεν αφήνει μεγάλα περιθώρια. Κάπως έτσι, καταλήξαμε ότι η συχνότητα παραγωγής δειγμάτων θα είναι τα 32258KHz. Αυτός ο ρυθμός είναι αισθητά μικρότερος από τα 44100Hz, που συναντάμε ως "χαμηλή" ρύθμιση στις σύγχρονες κάρτες. Ταυτόχρονα όμως είναι απόλυτα επαρκής για τους χοντροκομμένους ήχους που θα παράγει το δικό μας synthesizer. Εξάλλου, με την ίδια συχνότητα (32258KHz) θα πρέπει να παράγονται τέσσερα δείγματα –ένα για κάθε κανάλι–, γεγονός που ζορίζει τα πράγματα ακόμα περισσότερο. Παρεμπιπτόντως, νομίζουμε ότι τώρα κατανοείτε γιατί επιλέξαμε να γράψουμε τον κώδικα σε Assembly. Όπως και στην περίπτωση της δημιουργίας των scan lines από την κάρτα γραφικών, η παραγωγή των δειγμάτων απαιτεί όσο το δυνατόν πιο σταθερή συχνότητα.

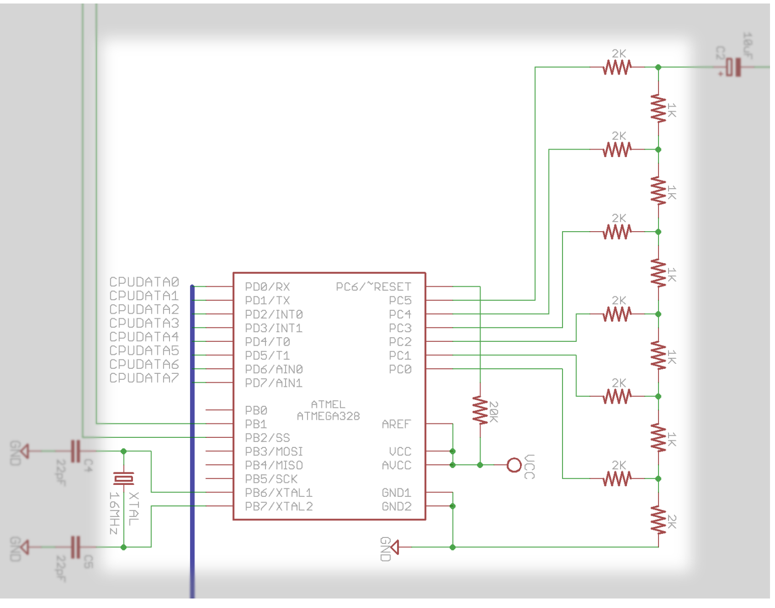


Ένας emulator του θρυλικού SID. Όπως μπορείτε να αντιληφθείτε κι από την εικόνα, το SID διαθέτει τρία κανάλια και στο καθένα μπορεί να δημιουργήσει τριγωνικά, πριονωτά ή τετραγωνικά κύματα, όπως επίσης και θόρυβο. Το SID είναι υβριδικό ολοκληρωμένο κι αυτό σημαίνει ότι εκτός από το ψηφιακό τμήμα, διαθέτει και αναλογικό. Στο δεύτερο βρίσκονται διάφορα φίλτρα και ένας πρόσθετος ταλαντωτής χαμηλής συχνότητας (Low Frequency Oscillator ή LFO), που χρησιμοποιείται στα εφέ. Ολόκληρο συγκρότημα, δηλαδή! (Σ.τ.Ε. Ποιος θυμάται αυτή την ατάκα;)

Ακόμα περισσότεροι ήχοι

Τη δεκαετία του '80, εκείνοι που συνέθεταν μουσική για τα παιχνίδια αναζητούσαν διαρκώς νέους ήχους. Η όρεξή τους ήταν αδύνατο να ικανοποιηθεί με ένα τριγωνικό κύμα και μερικά τετραγωνικά, που διαφοροποιούνταν μόνο ως προς το duty cycle. Στο κάτω κάτω, τα ίδια τα παιχνίδια απαιτούσαν ζωηρές μελωδίες, με πολλές και πλούσιες εναλλαγές στον ήχο, ικανές να εκφράσουν οτιδήποτε μπορεί να έκανε (ή να πάθαινε) ο ήρωας. Η διαρκής ανάγκη για νέους ήχους οδήγησε στη δημιουργία των πρώτων εφέ. Η ιδέα ήταν απλή: Για τη δημιουργία ενός νέου ήχου, αρκεί να πάρουμε έναν παλιό και ν' αρχίσουμε να μεταβάλλουμε κάποιο από τα χαρακτηριστικά του με μεγάλη ταχύτητα. Συνήθως, αυτή η "μεγάλη ταχύτητα" ήταν της τάξης των 100Hz. Ας δούμε μερικά από τα εφέ που υλοποιούνταν κατ' αυτόν τον τρόπο:

- **Tremolo:** Το πλάτος του κύματος πραγματοποιεί μια ταλάντωση και η ένταση του ήχου αυξομειώνεται ταχύτατα και διαρκώς.
- **Vibrato:** Σε αυτή την περίπτωση δεν ταλαντώνεται το πλάτος αλλά η συχνότητα του κύματος.
- **Glide & Portamento:** Αυτά τα δύο εφέ επιδρούν και πάλι στη συχνότητα. Αυτή τη φορά όμως δεν πραγματοποιείται κάποια ταλάντωση. Η εκάστοτε νότα ξεκινά κανονικά και, λίγο πριν ολοκληρωθεί, η συχνότητα του ήχου αρχίζει να αυξάνεται (pitch-bend up) ή να μειώνεται (pitch-bend down). Όταν αυτή η μεταβολή πραγματοποιείται στο ξεκίνημα της νότας, το εφέ έχει τη γενική ονομασία glide, ενώ στην πρώτη περίπτωση έχει το όνομα portamento.



Το κύκλωμα της κάρτας ήχου. Όλη η δουλειά γίνεται στο firmware του μικροελεγκτή, που τον μετατρέπει σε synthesizer. Το δίκτυο αντιστάσεων αποτελεί τον DAC που μετατρέπει τις ψηφιακές τιμές σε τάσεις και επιτρέπει την τελική σύνθεση του ηλεκτρικού κύματος.

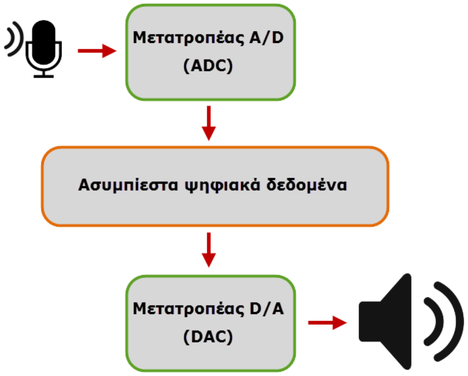
Μουσική αριθμητική

Προηγουμένως ασχοληθήκαμε με το ρυθμό παραγωγής δειγμάτων, που αποτελεί θεμελιώδες μέγεθος για ένα synthesizer. Ωστόσο, οι χρήστες ενός μουσικού οργάνου ενδιαφέρονται για έναν εντελώς διαφορετικό ρυθμό: αυτόν της μουσικής. Τώρα, μπορεί να νομίζετε ότι προσπαθούμε να μπλέξουμε δύο άσχετα πράγματα στην κουβέντα κι εν μέρει έχετε δίκιο. Ο ρυθμός της μουσικής και ο ρυθμός δειγματοληψίας αποτελούν δυο εντελώς ξένες έννοιες. Ωστόσο, σε ό,τι αφορά στη λειτουργία ενός synthesizer σε χαμηλό επίπεδο, τα συγκεκριμένα δύο μεγέθη δεν είναι άσχετα! Το ένα χτίζεται πάνω στο άλλο κι αυτό το διαπιστώνει κανείς πολύ γρήγορα, όταν προσπαθεί να υλοποιήσει ένα synthesizer από το μηδέν.

Όταν περιγράφαμε τις δυνατότητες της κάρτας ήχου, αναφέραμε ότι μπορεί να παίζει διάφορες νότες κατά παραγγελία. Αυτό σημαίνει ότι το firmware της κάρτας ήχου πρέπει να ενσωματώνει στοιχειώδεις γνώσεις μουσικής. Αν αναρωτιέστε γιατί, σκεφτείτε το εξής ερώτημα: Πόσο πρέπει να διαρκεί κάθε νότα; Από τα μαθήματα μουσικής του Γυμνασίου (δεν χρειάζεται δίπλωμα από ωδείο) γνωρίζουμε ότι η διάρκεια μιας νότας (ενός μουσικού φθόγγου, για την ακρίβεια) ονομάζεται και αξία. Γνωρίζουμε επίσης ότι οι τιμές που μπορεί να δεχτεί η αξία είναι το τέταρτο, το μισό, το ολόκληρο, το όγδοο, το δέκατο έκτο κ.λπ. Καλά όλα αυτά, θα πείτε, αλλά σε πόσο χρόνο αντιστοιχεί κάθε αξία; Η θεωρία απαντά και σ' αυτό: Η αξία του ενός τετάρτου αντιστοιχεί σε έναν κτύπο του tempo, το μισό διαρκεί δύο κτύπους, το ολόκληρο εκτείνεται σε τέσσερεις κτύπους, το όγδοο διαρκεί για μισό κτύπο και πάει λέγοντας. Μάλιστα! Μόλις προσθέσαμε ένα ακόμα επίπεδο αφαίρεσης και απάντηση δεν πήραμε. Έχετε δίκιο, αλλά αν κάνετε λίγη υπομονή ακόμα θα απαντηθούν όλα. Τι ακριβώς είναι αυτό το tempo;

Το tempo καθορίζει το βηματισμό μιας μελωδίας ή, αν προτιμάτε, την ταχύτητά της. Μπορούμε να το φανταζόμαστε σαν ένα ρολόι που παράγει κτύπους σε τακτά χρονικά διαστήματα και βοηθά τους μουσικούς να κρατούν ένα σταθερό ρυθμό. Οι μουσικοί χρησιμοποιούν διάφορους όρους για να χαρακτηρίζουν αυτή την ταχύτητα (Allegro, Adagio κ.ά.), αλλά υπάρχει και μια μονάδα μέτρησης που μπορεί να γίνει αντιληπτή από όλους. Ουσιαστικά, πρόκειται για το πλήθος των κτύπων ανά λεπτό (BPM από το Beats Per Minute). Ας επιστρέψουμε τώρα στο θέμα μας. Όπως είπαμε, η χρονική διάρκεια μιας νότας καθορίζεται από την αξία της κι αυτή από το tempo. Προφανώς, αυτά τα δύο μεγέθη επιλέγονται από τον συνθέτη της μελωδίας και όχι από το μουσικό όργανο. Το synthesizer αρκεί να περιλαμβάνει έναν μετρονόμο: Ένα χρονόμετρο που παράγει κτύπους, σύμφωνα με το επιλεγμένο tempo. Η διάρκεια κάθε νότας θα καθορίζεται από τους κτύπους αυτού του χρονομέτρου και την επιλεγμένη αξία.

Τα παραπάνω απαντούν στο ερώτημα για τη διάρκεια κάθε νότας, αλλά δεν τεκμηριώνουν τον αρχικό μας ισχυρισμό. Από πού κι ως πού συνδέεται ο ρυθμός της μουσικής με τον ρυθμό παραγωγής δειγμάτων; Προηγουμένως είπαμε ότι το synthesizer πρέπει να λειτουργεί και σαν χρονόμετρο. Μήπως αυτό σημαίνει ότι πρέπει να γράψουμε κώδικα για την υλοποίηση ενός ρολογιού; Κάτι τέτοιο θα ήταν εντελώς περιττό. Το synthesizer ενσωματώνει ήδη έναν μηχανισμό, ο οποίος ενεργοποιείται σε τακτά χρονικά διαστήματα και με τεράστια ακρίβεια. Αναφερόμαστε στο



Ο ήχος μετατρέπεται σε "ηλεκτρικό κύμα" με τη βοήθεια ενός μικροφώνου. Το εν λόγω κύμα οδηγείται σε έναν ADC (Analog to Digital Converter), ο οποίος λαμβάνει δείγματα με μια συχνότητα που ονομάζεται συχνότητα δειγματοληψίας. Τα δείγματα αποτελούν την ψηφιακή αναπαράσταση του αρχικού ήχου – σε ασυμπίεστη μορφή. Για την αναπαραγωγή του ψηφιακού ήχου χρησιμοποιείται ένας DAC (Digital to Analog Converter). Στην έξοδό του παράγονται οι τάσεις που είχαν "δειγματιστεί" και σιγά-σιγά σχηματίζεται το αρχικό ηλεκτρικό κύμα. Η μέθοδος DDS (Direct Digital Synthesis) για τη σύνθεση ήχου προβλέπει την απομάκρυνση της πρώτης βαθμίδας και την κατασκευή των δειγμάτων με μαθηματικό τρόπο.

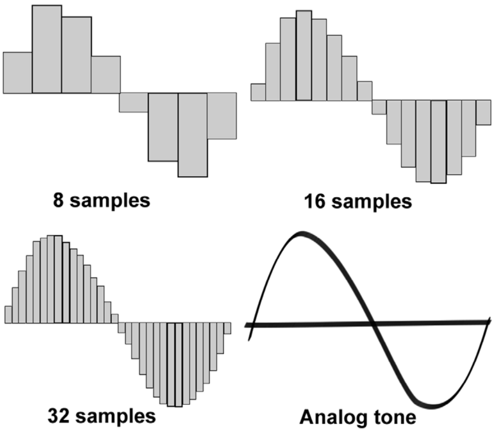
μηχανισμό που παράγει τα δείγματα. Τελικά, για τη διατήρηση του μουσικού ρυθμού ο μετρονόμος του synthesizer μπορεί να μετράει το χρόνο με βάση το πλήθος των δειγμάτων που έχουν παραχθεί.

Μέτρηση χρόνου

Η κάρτα ήχου που σχεδιάσαμε πλαισιώνει ένα σύστημα που μοιάζει με τους υπολογιστές του '80. Ήμασταν υποχρεωμένοι, λοιπόν, να εξασφαλίσουμε ότι θα μπορεί να παίξει όλα τα σχετικά κομμάτια. Τα παιχνίδια εκείνης της εποχής (από εκεί προέρχονται οι πιο γνωστές μελωδίες) συνοδεύονταν από μουσική με καταιγιστικό ρυθμό. Το πιο διαδεδομένο tempo ήταν τα 180BPM και ακολούθως τα 150BPM, ενώ ορισμένες φορές επιλέγονταν τα 120BPM και σπανιότερα τα 90BPM. Περιττό να πούμε ότι το synthesizer που σχεδιάσαμε υποστηρίζει αυτούς τους ρυθμούς και μόνο. Επιπρόσθετα, το synthesizer μπορεί να παίζει νότες με ελάχιστη διάρκεια το ένα τριακοστό δεύτερο (ένα όγδοο του κτύπου), καθώς οι σχετικές μελωδίες δεν περιλάμβαναν νότες με μικρότερη αξία. Όπως βλέπετε, το πρόγραμμα υποστηρίζει τέσσερεις διαφορετικές συχνότητες για τους κτύπους του tempo, ενώ για τη διάρκεια κάθε νότας πρέπει να διακρίνει και υποδιαιρέσεις των κτύπων. Χρειαζόμαστε ένα περίπλοκο χρονόμετρο... ή μήπως όχι;

Σχεδιάζοντας το μετρονόμο αντιληφθήκαμε γρήγορα ότι πρέπει να μετράει το χρόνο σε υποδιαιρέσεις του κτύπου κι όχι σε ολόκληρους κτύπους. Επιπρόσθετα, αποφασίσαμε ότι αυτές οι υποδιαιρέσεις θα έχουν σταθερή συχνότητα. Με αυτόν τον τρόπο, ο κώδικας του μετρονόμου απλοποιήθηκε αρκετά και την ίδια στιγμή το synthesizer απέκτησε μεγάλη ευελιξία: Για να αλλάξουμε το ρυθμό αναπαραγωγής, αρκεί να αλλάξουμε την αντιστοιχία μεταξύ κτύπων και υποδιαιρέσεων του κτύπου. Μην ανησυχείτε αν αυτή η περιγραφή σας προκάλεσε σύγχυση. Θα εξετάσουμε δυο παραδείγματα και θα καταλάβετε αμέσως τι εννοούμε. Η συχνότητα που επιλέξαμε για τις υποδιαιρέσεις των κτύπων είναι τα 192Hz. Στο ρυθμό των 90BPM, οι κτύποι επαναλαμβάνονται με συχνότητα 1,5Hz. Επομένως, όταν επιλέγεται ο συγκεκριμένος ρυθμός, ο κώδικας του μετρονόμου θεωρεί ότι ένας κτύπος αντιστοιχεί σε 128 υποδιαιρέσεις (192Hz / 128 = 1,5Hz). Ας δούμε και την περίπτωση που επιλέγονται τα 192BPM. Σε αυτό το ρυθμό οι κτύποι επαναλαμβάνονται με συχνότητα 3Hz.

Αυτή τη φορά, ο μετρονόμος του synthesizer θεωρεί ότι ένας κτύπος αντιστοιχεί σε 64 υποδιαιρέσεις (192Hz / 64 = 3Hz). Με παρόμοιο τρόπο –σαν πολλαπλάσιο των υποδιαιρέσεων του κτύπου–, ορίσαμε και όλες τις αξίες (διάρκειες) για κάθε υποστηριζόμενο ρυθμό. Έτσι, κάθε φορά που αλλάζει το tempo, το μόνο που διαφοροποιείται στον κώδικα είναι ένας πίνακας με τις αντιστοιχίες ανάμεσα σε διάρκειες και υποδιαιρέσεις του κτύπου.



Όσο μεγαλύτερη είναι η συχνότητα δειγματοληψίας, τόσο πιο πιστή είναι η αναπαράσταση του ηλεκτρικού κύματος. Στην περίπτωση της σύνθεσης, όμως, η μεγάλη "πυκνότητα" δειγμάτων δεν μπορεί να εγγυηθεί την ποιότητα. Ο σημαντικότερος παράγοντας αφορά στην κατασκευή των δειγμάτων. Αν υπάρχουν σφάλματα εκεί, το τελικό κύμα θα περιέχει παράσιτα.

Αναπαραγωγή

Κάπου εδώ είναι πιθανό να μπουχτίσατε με τις ατελείωτες αναλύσεις. Πότε θα περάσουμε στο ψητό και θα μιλήσουμε για τον κώδικα; Ε, λοιπόν, η συζήτηση για την ουσία του κώδικα έχει ξεκινήσει προ πολλού. Περιττό να πούμε ότι με όσα προηγήθηκαν μπορείτε να κατανοήσετε το πιο σημαντικό τμήμα του προγράμματος: Τη συνάρτηση που παίζει μουσική! Όταν λειτουργεί ο υπολογιστής, ο χρήστης μπορεί να περιγράφει μικρές μελωδίες και να τις αντιστοιχεί στα διαθέσιμα κανάλια. Προφανώς, κάθε μελωδία απαρτίζεται από νότες. Για κάθε νότα, ο χρήστης δηλώνει οκτάβα, διάρκεια και, αν το επιθυμεί, εφέ. Το πώς επιτυγχάνονται όλα αυτά δεν θα μας απασχολήσει τώρα. Σημασία έχει ότι όταν ζητάμε από την κάρτα ήχου (το synthesizer) να παίξει την αποθηκευμένη μελωδία, οι νότες για κάθε κανάλι βρίσκονται ήδη στη μνήμη, η μία μετά την άλλη. Η αναπαραγωγή πραγματοποιείται από τη συνάρτηση ονόματι play (περιμένατε άλλη ονομασία;) που βρίσκεται στο ομώνυμο αρχείο (play.asm).

Η συνάρτηση play ξεκινά με την κλήση της init_music_data, η οποία αρχικοποιεί μια σειρά βοηθητικών μεταβλητών. Για την ώρα μπορείτε να την αγνοήσετε. Η αναπαραγωγή αυτή καθ' αυτή πραγματοποιείται στο βρόχο με το εμπνευσμένο όνομα play_loop. Εκ πρώτης ο κώδικας μοιάζει ακατανόητος, αλλά δεν κάνει τίποτα διαφορετικό απ' όσα έχουμε πει. Αρχικά υπολογίζονται 166 νέα δείγματα, μέσα σ' έναν μικρότερο βρόχο που φροντίζει να εισάγει τις απαραίτητες καθυστερήσεις.

```
sample_loop:
      rcall samples
      ldi tmp3, 116
      rcall delay
```

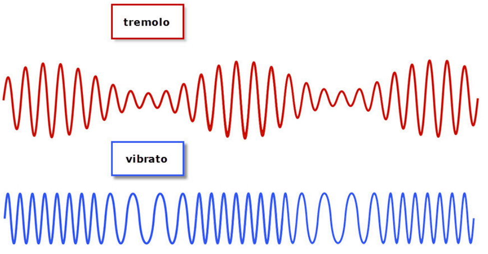
```
nop
nop
dec loop_cnt
brne sample_loop
nop
```

Οι καθυστερήσεις είναι κατάλληλα υπολογισμένες, ώστε τα δείγματα να παράγονται με την επιθυμητή συχνότητα των 32258Hz. Ακολουθεί ο υπολογισμός ενός δείγματος και αμέσως μετά καλείται η συνάρτηση update, μία φορά για καθένα από τα τρία πρώτα κανάλια:

```
rcall samples
ldi channel_data, ch1_data
rcall update
ldi channel_data, ch2_data
rcall update
ldi channel_data, ch3_data
rcall update
nop
nop
ldi tmp3, 22
rcall delay
```

Αυτή τη φορά εισάγονται μικρότερες καθυστερήσεις, αφού ο χρόνος μέχρι το επόμενο δείγμα "γεμίζει" με την εκτέλεση της update εις τριπλούν. Το πρόγραμμα συνεχίζεται με τον υπολογισμό ενός ακόμα δείγματος, την κλήση της update για το τέταρτο κανάλι, καθώς και δύο ελέγχους. Ουσιαστικά, τσεκάρουμε αν εξαντλήθηκαν οι νότες για όλα τα κανάλια, καθώς και το αν ο επεξεργαστής έστειλε την εντολή διακοπής. Και στις δύο αυτές περιπτώσεις, η αναπαραγωγή πρέπει σταματά. Διαφορετικά, ο βρόχος play_loop εκτελείται από την αρχή. Εννοείται ότι η κλήση της update (για το τέταρτο κανάλι) και οι έλεγχοι που περιγράψαμε διαρκούν ακριβώς όσο πρέπει, ώστε όταν συνεχίζεται η αναπαραγωγή το επόμενο δείγμα να υπολογίζεται στον προβλεπόμενο χρόνο (με συχνότητα 32258Hz).

Αναρωτιέστε που πήγαν όλα αυτά που λέγαμε για τις υποδιαιρέσεις των κτύπων και την τήρηση του επιλεγμένου ρυθμού; Κάθε φορά που υπολογίζονται 168 δείγματα, ο κώδικας που εξετάσαμε εκτελεί τη συνάρτηση update για κάθε κανάλι. Ξέρουμε επίσης ότι τα δείγματα παράγονται με συχνότητα 32258Hz. Επομένως, η συνάρτηση update εκτελείται για κάθε κανάλι με συχνότητα 32258Hz / 168. Αυτή η διαίρεση μας δίνει 192.002Hz. Σας θυμίζει τίποτα αυτό το νούμερο; Είναι η συχνότητα επανάληψης των υποδιαιρέσεων των κτύπων. Ας ρίξουμε μια ματιά στη συνάρτηση



Πολύ συχνά, οι όροι tremolo και vibrato χρησιμοποιούνται αδιακρίτως. Στην πραγματικότητα πρόκειται για δύο εντελώς διαφορετικά εφέ. Το tremolo προβλέπει την περιοδική μεταβολή του πλάτους, ενώ το vibrato αναφέρεται στην περιοδική μεταβολή της συχνότητας.

update (βρίσκεται στο ομώνυμο αρχείο – καμία πρωτοτυπία). Αρχικά, φορτώνονται όλες οι παράμετροι για τη νότα που αναπαράγεται στο εκάστοτε κανάλι. Μεταξύ άλλων, θα δείτε ότι φορτώνεται και ένα μέγεθος που το ονομάζουμε duration. Πρόκειται για τη διάρκεια της νότας, εκφρασμένη σε υποδιαιρέσεις του κτύπου. Κάθε φορά που εκτελείται η συνάρτηση update (δηλαδή σε κάθε υποδιαίρεση του κτύπου), η προαναφερθείσα τιμή μειώνεται. Όταν μηδενιστεί, θεωρούμε ότι η διάρκεια της νότας έληξε και ο κώδικας φροντίζει να φορτώσει την επόμενη νότα. Με τη λήψη της νέας νότας, το μέγεθος duration λαμβάνει μια νέα τιμή που εκφράζει, σε υποδιαιρέσεις του κτύπου, τη συνολική αξία της νότας. Στο τέλος της ίδιας συνάρτησης (update) εφαρμόζεται και το όποιο εφέ. Αυτό σημαίνει ότι η τροποποίηση των παραμέτρων που προβλέπει κάθε εφέ, πραγματοποιείται με συχνότητα 192Hz.

Εύλογο ερώτημα…

Αν εξακολουθείτε να έχετε ερωτηματικά για το πώς διατηρείται ο εκάστοτε ρυθμός και πώς εκφράζονται κάθε φορά οι αξίες, θα σας προτείναμε να ρίξετε μια ματιά στο αρχείο melody.asm. Στις γραμμές 102 έως 108 ορίζουμε έναν πίνακα, ο οποίος εκφράζει όλες τις αξίες των φθόγγων σε υποδιαιρέσεις του κτύπου, σε κάθε ρυθμό. Μελετώντας όμως τον πίνακα ενδέχεται να δημιουργηθεί μια καινούργια απορία. Θα περίμενε κανείς να έχουμε ορίσει με τέτοιον τρόπο τις υποδιαιρέσεις του κτύπου, ώστε μία υποδιαίρεση να αντιστοιχεί στη μικρότερη αξία (1/32) του γρηγορότερου ρυθμού (180BPM). Αφού αυτή η είναι μικρότερη διάρκεια που μπορεί να έχουμε σε οποιαδήποτε περίπτωση, γιατί δεν την αντιστοιχίσαμε σε μία μόνο υποδιαίρεση; Αν και αυτό θα έμοιαζε καλή ιδέα, υπάρχουν τουλάχιστον δύο λόγοι για τους οποίους θα μας δημιουργούσε προβλήματα. Κατ' αρχάς, αν στη μικρότερη αξία αντιστοιχούσαμε μια υποδιαίρεση και όχι οκτώ, θα έπρεπε να μειώσουμε τη συχνότητα των υποδιαιρέσεων κατά οκτώ φορές. Αυτό σημαίνει ότι η συνάρτηση update θα έπρεπε να καλείται 24 φορές το δευτερόλεπτο και όχι 192, γεγονός που θα την καθιστούσε ακατάλληλη για την εφαρμογή των εφέ. Άρα το πρόγραμμα θα έπρεπε να εκτελεί την update κάθε Χ δείγματα και μια ξεχωριστή συνάρτηση για τα εφέ κάθε Υ δείγματα. Εν ολίγοις, ο βρόχος play_loop θα περιπλεκόταν αρκετά. Ο δεύτερος λόγος είναι ακόμα πιο ζόρικος στην αντιμετώπιση. Μπορεί να μην έχουμε πει τίποτα μέχρι στιγμής, αλλά το μικρό μας synthesizer εφαρμόζει σε κάθε νότα ένα υποτυπώδες περίγραμμα έντασης (volume envelope ή volume contour). Στο ξεκίνημα κάθε νότας πραγματοποιείται

μια μικρή αύξηση της έντασης και στο τελείωμα μια σύντομη μείωση. Έτσι οι νότες διακρίνονται καλύτερα και δεν μοιάζουν μ' ένα συνεχόμενο σφύριγμα. Μπορείτε να διαβάσετε περισσότερα για τα περιγράμματα έντασης στο άρθρο "Arduino Sound Master: Κύμα μέσα στο κύμα" (τεύχος 017). Αν στα 180BPM οι νότες με αξία 1/32 διαρκούσαν ακριβώς μια υποδιαίρεση του κτύπου, η συνάρτηση update θα εκτελούνταν μόνο μια φορά για κάθε τέτοια νότα και δεν θα προλάβαινε να υλοποιήσει το περίγραμμα έντασης. Εδώ που τα λέμε, το ίδιο πρόβλημα θα προέκυπτε και για τις λίγο μεγαλύτερες αξίες, όπως και για παραπλήσιες αξίες σε μικρότερα tempo. Μπορεί το περίγραμμα που έχουμε υλοποιήσει να είναι απλούστατο, αλλά ο σχηματισμός του απαιτεί οκτώ βήματα. Επομένως, η μικρότερη αξία (η πιο σύντομη νότα) έπρεπε να εκτείνεται σε τουλάχιστον οκτώ υποδιαιρέσεις του κτύπου.

Σύνθεση κύματος

Ο τίτλος της παραγράφου ενδέχεται να προκαλεί σύγχυση. Μα, δεν το εξαντλήσαμε αυτό το θέμα; Η αλήθεια είναι πως όχι. Συζητήσαμε για το ρυθμό των δειγμάτων, αλλά δεν είπαμε κουβέντα για το πώς παράγονται. Ο υπολογισμός μερικών αριθμητικών τιμών που σχηματίζουν ένα τριγωνικό ή ένα τετραγωνικό κύμα δεν αποτελεί μεγάλη πρόκληση. Στην πρώτη περίπτωση, οι τιμές πρέπει αρχικά να ακολουθούν μια ανοδική πορεία και στη συνέχεια μια μειούμενη, πάντα με τον ίδιο ρυθμό. Στη δεύτερη περίπτωση αρκούν μερικές τιμές σε μια συγκεκριμένη στάθμη και, ακολούθως, λίγες ακόμα σε μιαν άλλη στάθμη. Η πρόκληση δεν εντοπίζεται στους υπολογισμούς, αλλά στο πλήθος τους: Πόσα δείγματα πρέπει να υπολογίζουμε σε κάθε περίοδο του κύματος; Αν υπολογίζουμε πάρα πολλά, το κύμα θα μοιάζει "απλωμένο" και η συχνότητά του θα είναι χαμηλή. Αντιθέτως, αν υπολογίζουμε λίγα, το κύμα θα μοιάζει "συμπυκνωμένο" και η συχνότητά του θα είναι αυξημένη. Μπερδευτήκατε; Ας το πάρουμε αλλιώς.

Ο ρυθμός με τον οποίο παράγονται τα δείγματα είναι σταθερός (32258Hz). Με τον ίδιο ρυθμό τροφοδοτούνται στον DAC και μετατρέπονται σε τάσεις. Πώς μπορούμε να σχηματίσουμε ένα κύμα με συχνότητα 440Hz; Για να πετύχουμε κάτι τέτοιο, πρέπει να υπολογίζουμε 73 δείγματα για κάθε περίοδο. Προσπεράστε την επιλογή του 73 και σκεφτείτε αυτό: Αν κάθε δείγμα παράγεται με συχνότητα 32258Hz, τότε κάθε ομάδα των 73 δειγμάτων παράγεται με συχνότητα 32258Hz / 73. Με άλλα λόγια, κάθε περίοδος του κύματος που συνθέτουμε θα παράγεται με συχνότητα 441,8Hz. Μπορεί να μην πετύχαμε τα 440Hz με εντυπωσιακή ακρίβεια, αλλά κανένα αφτί δεν θα ακούσει τη διαφορά :) Νομίζουμε ότι τώρα καταλαβαίνετε πώς προέκυψε το 73: Διαιρέσαμε τη συχνότητα παραγωγής των δειγμάτων με τη συχνότητα που θέλαμε να σχηματίσουμε.

Τελικά, ούτε κι αυτό το πρόβλημα μοιάζει δύσκολο. Κάθε φορά που πρέπει να συνθέσουμε μια νότα, αρκεί να διαιρέσουμε το 32258 με τη συχνότητα της νότας. Το αποτέλεσμα της διαίρεσης αντιστοιχεί στο πλήθος των δειγμάτων που πρέπει να υπολογίζουμε ανά περίοδο. Χρήσιμη διαπίστωση, αλλά το πρόβλημα δεν λύνεται τόσο εύκολα. Τώρα που ξέρουμε πόσα δείγματα χρειαζόμαστε, πρέπει να βρούμε κι έναν τρόπο να τα υπολογίσουμε. Βασικά, πρέπει να διαιρέσουμε την



Ένας κλασικός αναλογικός μετρονόμος, με κουρδιστήρι. Όταν βρίσκεται σε χρήση, το διαφανές καπάκι της πρόσοψης απομακρύνεται και η μεταλλική ράβδος πραγματοποιεί ταλαντώσεις. Κάθε φορά που φτάνει σε κάποιο άκρο, ακούγεται ένας κτύπος. Μετακινώντας το μικρό βαρίδι μπορούμε να ρυθμίσουμε τον επιθυμητό ρυθμό.

περίοδο του εκάστοτε κύματος σε τόσα τμήματα, όσα είναι και τα απαιτούμενα δείγματα. Για το σκοπό αυτό μπορούμε να επιστρατεύσουμε έναν μετρητή, που θα δείχνει σε ποιο στάδιο της περιόδου βρισκόμαστε. Ο συγκεκριμένος μετρητής μπορεί να παρομοιαστεί με τη φάση ενός αρμονικού κύματος και, εδώ που τα λέμε, στην ορολογία των synthesizer ονομάζεται ακριβώς έτσι: φάση (phase). Ας επιστρέψουμε στο πρόβλημα όμως. Για να χωρίσουμε την περίοδο σ' ένα αυθαίρετο πλήθος τμημάτων, αρκεί να ορίσουμε το κατάλληλο βήμα προσαύξησης για τη φάση. Για παράδειγμα, αν χρειαζόμαστε Κ δείγματα ανά περίοδο και ο μετρητής (η φάση) κυμαίνεται στο διάστημα από 0 (αρχή της περιόδου) έως Ζ (τέλος της περιόδου), αρκεί να χωρίσουμε το μέγεθος Ζ σε Κ ίσα μέρη. Το αποτέλεσμα αυτής της διαίρεσης θα μας δώσει το βήμα προσαύξησης της φάσης, που πολύ συχνά αποκαλείται και phase delta. (Ψαρωτικό, δεν μπορείτε να πείτε.) Ο κώδικας, λοιπόν, για να συνθέσει την περίοδο ενός κύματος, πρέπει να κάνει τα εξής: Αρχικά μηδενίζει τον μετρητή που αντιστοιχεί στη φάση. Ακολούθως, κάθε φορά που πρέπει να υπολογιστεί ένα δείγμα, προσθέτει στη φάση το phase delta. Έτσι προκύπτει η νέα τιμή της φάσης, σύμφωνα με την οποία παράγεται το νέο δείγμα. Όταν η φάση φτάσει στη μέγιστη τιμή, η περίοδος του κύματος θεωρείται ολοκληρωμένη και η διαδικασία ξεκινά από την αρχή, για το σχηματισμό της επόμενης περιόδου. Αν ανοίξετε το αρχείο melody.asm, θα δείτε ότι δημιουργούμε στη μνήμη προγράμματος έναν πίνακα τιμών με το όνομα deltas (γραμμές 5 έως 83). Σε αυτόν τον πίνακα έχουμε αποθηκεύσει όλες τις τιμές του phase delta για κάθε νότα, για κάθε οκτάβα. Έτσι, όταν η κάρτα ήχου πρέπει να συνθέσει μια νότα, δεν χρειάζεται να υπολογίσει το απαιτούμενο πλήθος δειγμάτων, ούτε το βήμα προσαύξησης της φάσης. Όλες αυτές οι πράξεις έχουν γίνει από εμάς και το πρόγραμμα μπορεί να καταπιαστεί με την ουσία: τη σύνθεση του κύματος δείγμα προς δείγμα.

Μη νομίζετε ότι η μελέτη της κάρτας ήχου τελειώνει εδώ. Μπορεί να εξετάσαμε τα πιο ζόρικα τμήματα του κώδικα, αλλά δεν έχουμε πει τίποτα για τη γενική διάρθρωση του προγράμματος. Στο επόμενο τεύχος θα ολοκληρώσουμε τη μελέτη και θα εξετάσουμε ένα πολύ πιο διασκεδαστικό ζήτημα: Την κατασκευή ενός προγράμματος που διευκολύνει τους μουσικούς να συνθέτουν αριστουργήματα, ειδικά για την κάρτα μας :D

DigitalOcean

Ταχύτατα VPSes στο cloud, σε hosts με δίσκους SSD.
Επιλογή datacenter σε Ευρώπη, Αμερική και Ασία.
Lean-mean control panel, για απόλυτο έλεγχο.

Αποκτήστε τώρα το δικό σας VPS,
στο cloud της DigitalOcean.
Κάντε κλικ στο http://bit.ly/digocean10off
και κερδίστε αυτομάτως 10$ σε credit.

Hint: Επιλέγοντας το μικρό πλάνο,
πάντα με κλικ στο http://bit.ly/digocean10off,
ουσιαστικά έχετε δύο μήνες δωρεάν για ένα VPS
με 512MB RAM, 20GB SSD και 1TB transfer.
Δεν είναι κι άσχημα

;)

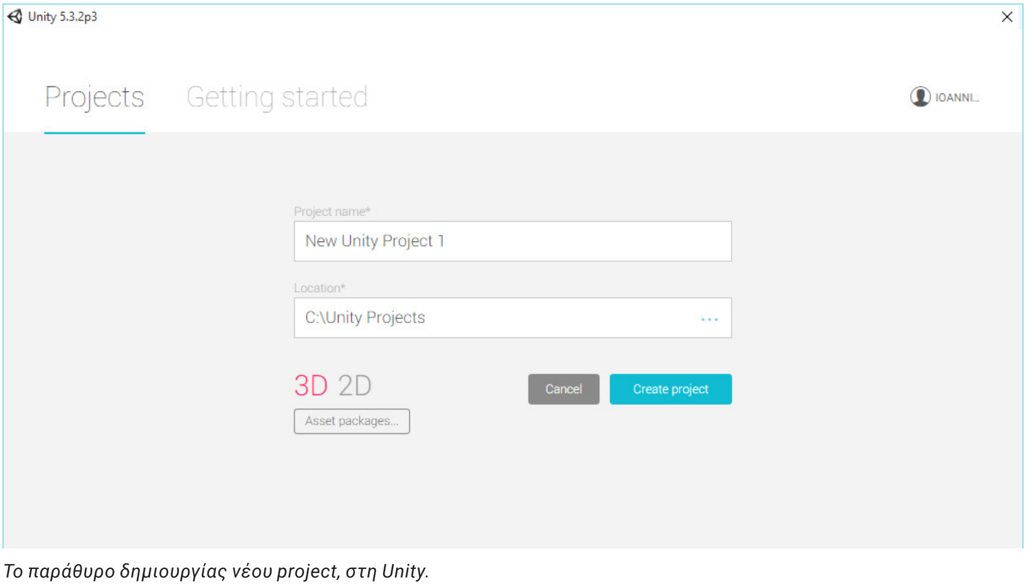
Skill: Intermediate
Tags: Indie games, Unity, development

Η μηχανή δημιουργίας παιχνιδιών Unity

Στο προηγούμενο άρθρο της σειράς, που αρχίζει από τη σελίδα 14, κάναμε μια αναφορά στη μηχανή δημιουργίας παιχνιδιών Unity. Δεν θα αναφερθούμε στη διαδικασία της εγκατάστασης της μηχανής. Για προγραμματιστές σαν εμάς, θα λέγαμε ότι η εγκατάσταση ενός προγράμματος ήταν δύσκολη υπόθεση προ αμνημονεύτων ετών – ή μήπως ποτέ; Πριν πάντως ξεκινήσουμε να προγραμματίζουμε το παιχνίδι μας, καλό θα ήταν να εξοικειωθούμε λίγο με το πρόγραμμα, το περιβάλλον του καθώς και με τις ιδιαιτερότητες που ίσως έχει.

του Γιάννη Κρομμύδα

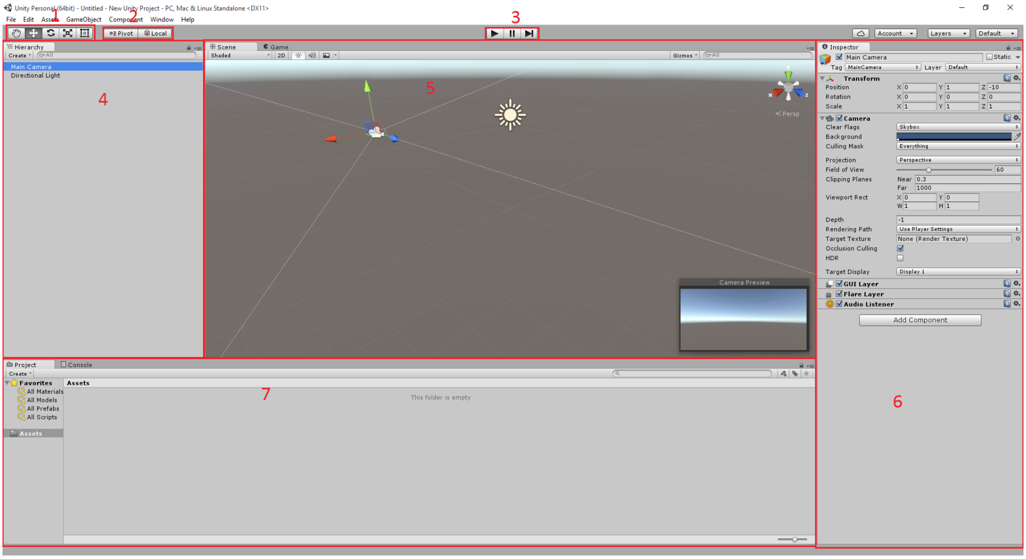
Με το που ξεκινάμε τη Unity, εμφανίζεται ένα παράθυρο που μας προτρέπει να επιλέξουμε να συνεχίσουμε να δουλεύουμε σε ένα υπάρχον project ή να δημιουργήσουμε ένα νέο. Η δημιουργία project δεν προϋποθέτει την ύπαρξη assets. Αρκεί να δηλώσουμε ένα path, κάτω από το οποίο θα δημιουργηθεί το folder structure που θα δηλώνει την ύπαρξη του project μας, καθώς και τον τύπο του (2D ή 3D). Η αλήθεια είναι ότι η επιλογή του τύπου του project δεν κάνει κάποια ουσιαστική διαφορά, μια που μέσα από τη μηχανή μπορούμε να περάσουμε από 2D mode σε 3D mode και το αντίστροφο, όποτε κι όσες φορές το θελήσουμε.



Το παράθυρο δημιουργίας νέου project, στη Unity.

Επιλέγοντας το κουμπί Asset Packages, η μηχανή θα μας δώσει τη δυνατότητα να κάνουμε import στο καινούργιο μας project πακέτα με assets που υπάρχουν στον υπολογιστή μας. Ακόμη και να μην έχουμε δουλέψει *ποτέ* ξανά με τη μηχανή στο παρελθόν, είναι πολύ πιθανό να δούμε ήδη εγκατεστημένα πακέτα στο σύστημά μας. Δεν υπάρχει λόγος ανησυχίας, βέβαια, μια κι αυτά έρχονται με την εγκατάσταση της μηχανής. Περισσότερα για τα πακέτα θα πούμε αργότερα.

Αμέσως μετά τη δημιουργία του project μας, ανοίγει το κύριο παράθυρο της μηχανής. Αυτό θα είναι το παράθυρο που θα κάνουμε το 99% της δουλειάς μας. Ας δούμε λεπτομερώς τα περιεχόμενα του παραθύρου, αναλύοντας ταυτόχρονα και μερικές έννοιες που χρησιμοποιεί η μηχανή.



Το παράθυρο εργασίας της μηχανής. Το μέγεθος και η θέση των επιμέρους εσωτερικών παραθύρων μπορούν να αλλάξουν ανά πάσα στιγμή και να προσαρμοστούν στα γούστα μας. Με τον καιρό είναι σίγουρο ότι θα βρεθεί κάποιο layout που θα μας διευκολύνει, ανάλογα βέβαια με τη δουλειά που κάνουμε. Σε καθεμία από τις αριθμημένες περιοχές θα εστιάσουμε την προσοχή μας στο υπόλοιπο του άρθρου.

Η πρώτη, καλή γνωριμία

Ας ξεκινήσουμε από το κέντρο της οθόνης, στην περιοχή 5 της εικόνας. Στο παράθυρο αυτό βλέπουμε τη σκηνή (scene). Εδώ έχουμε τη δυνατότητα να επιλέγουμε και να αλληλεπιδράσουμε με τα αντικείμενα (game objects) που έχουμε τοποθετήσει στη σκηνή μας. Στην πάνω δεξιά γωνία του παραθύρου μπορούμε πάντα να δούμε το λεγόμενο gizmo της σκηνής. Χάρη σ' αυτό παρατηρούμε τη σκηνή από ένα συγκεκριμένο άξονα (κάνοντας κλικ στο αντίστοιχο χρώμα του) και αλλάζουμε μεταξύ των δύο τύπων προβολής: προοπτικής και ορθογραφικής ή ισομετρικής (perspective και orthographic ή isometric αντίστοιχα). Η αλλαγή του τύπου προβολής γίνεται αν κάνουμε κλικ στον κύβο στη μέση του gizmo ή στο κείμενο κάτω από αυτό.

Στη μπάρα που υπάρχει στο πάνω μέρος του παραθύρου και από αριστερά προς τα δεξιά, βλέπουμε τα εξής:

- Από το dropdown που γράφει Shaded, επιλέγουμε τον τρόπο απεικόνισης (render) των game objects. Θα μας απασχολήσουν τρεις επιλογές:
- Shaded, για να βλέπουμε τις επιφάνειες του μοντέλου με τις υφές (textures) τους
- Wireframe, για να βλέπουμε τον σκελετό των μοντέλων
- Shaded Wireframe, μείξη των δύο προαναφερθέντων, στην οποία φαίνονται και ο σκελετός των μοντέλων και οι υφές τους.
- Στη συνέχεια παρατηρούμε το κουμπί το οποίο εναλλάσσει την οπτική της σκηνής από 3D σε 2D. Η εναλλαγή σε 2D ουσιαστικά μας βάζει να δούμε τη σκηνή από συγκεκριμένη οπτική γωνία και με την ορθογραφική επιλογή στην κάμερα. Σημειώστε αυτό το κουμπί γιατί θα το χρησιμοποιούμε για να δουλεύουμε στο UI των παιχνιδιών μας.
- Με το επόμενο κουμπί μπορούμε να ενεργοποιήσουμε ή να απενεργοποιήσουμε τα φώτα που έχουμε τοποθετήσει στη σκηνή μας. Αυτό μπορεί να φανεί χρήσιμο όταν θελήσουμε να κάνουμε αλλαγές σε μια πολύ σκοτεινή πίστα. (Η αλήθεια είναι ότι οι σκοτεινές πίστες στα παιχνίδια δεν στηρίζονται στην έλλειψη φωτισμού, αλλά σε πηγές "μαύρου φωτός").
- Το επόμενο κουμπί ενεργοποιεί ή απενεργοποιεί τις πηγές ήχου που υπάρχουν στη σκηνή μας.
- Κάνοντας κλικ στο τελευταίο κουμπί στην αριστερή ομάδα, μπορούμε να ενεργοποιήσουμε ή να απενεργοποιήσουμε το λεγόμενο skybox. Ουσιαστικά το skybox είναι ένα πολύ μεγάλο κουτί, στις επιφάνειες του οποίου έχουν προσκολληθεί (από τη μέσα μεριά) οι εικόνες που θα αντιπροσωπεύουν τον ουρανό της σκηνής μας. Ως gizmos ονομάζονται τα εικονίδια που εμφανίζονται στη σκηνή μας και δηλώνουν ότι, π.χ., ένα game object έχει ένα component τύπου light σ' αυτό (θα εμφανιστεί μια λάμπα με το χρώμα του φωτός, στο σημείο που βρίσκεται το φως στη σκηνή).
- Στην άλλη άκρη της μπάρας, συνεχίζουμε με το κουμπί που ενεργοποιεί και απενεργοποιεί την ορατότητα των gizmos στη σκηνή.
- Τέλος, η Unity μάς επιτρέπει να ψάξουμε για συγκεκριμένα αντικείμενα στη σκηνή μας μέσω του search box, στη δεξιά μεριά της μπάρας.

Στο ίδιο παράθυρο υπάρχει ακόμα ένα tab, ονόματι Game. Σε αυτό μπορούμε να δούμε τι δείχνει η κύρια κάμερα που υπάρχει στη σκηνή μας. Ουσιαστικά, δείχνει το τι θα βλέπει ο παίχτης όταν τρέχει το παιχνίδι.

Ας πάμε αριστερά, στο παράθυρο Hierarchy (περιοχή 4 στο screenshot μας). Εδώ υπάρχει μια ιεραρχική λίστα όλων των αντικειμένων που υπάρχουν στην σκηνή μας. Μπορούμε να τα επιλέξουμε και να τους αλλάξουμε θέση στη λίστα, κάνοντας drag and drop. Ένα game object δύναται να έχει άλλα game objects ως παιδιά του. Όταν

ισχύει αυτό, αριστερά από το όνομα του parent game object, εμφανίζεται ένα βελάκι που συμπτύσσει ή αναδιπλώνει τη λίστα με τα παιδιά (ή αλλιώς με τα game objects απογόνους). Ένα χρήσιμο dropdown στην αριστερή πάνω γωνία είναι το Create, που μας εμφανίζει μια λίστα από όλα τα game objects που μπορούν να προστεθούν στη σκηνή μας. Δίπλα του υπάρχει ένα search box για τα ονόματα των αντικειμένων που βρίσκονται σε αυτή τη λίστα.

Συνεχίζοντας κάτω, στην περιοχή 7 του screenshot και στην καρτέλα Project, βλέπουμε όλα τα assets που έχουν προστεθεί στο project μας. Αυτή τη στιγμή δεν έχουμε προσθέσει κάποιο για να εμφανιστεί. Έχουμε και εδώ ένα dropdown με όνομα Create, αυτό όμως έχει διαφορετική λειτουργία από εκείνο που είδαμε προηγουμένως. Από το συγκεκριμένο δημιουργούμε assets για το παιχνίδι μας, όπως, π.χ., νέα αρχεία κώδικα. Στα δεξιά της μπάρας έχουμε ακόμα ένα βολικό search box, για τα assets του project μας αυτή τη φορά.

Στο ίδιο παράθυρο και στην καρτέλα Console βλέπουμε την κονσόλα της μηχανής (Σ.τ.Ε. duh!), στην οποία μπορούμε να εκτυπώνουμε μέσα από τον κώδικά μας. Εδώ θα εμφανίζονται τυχόν λάθη που ίιιισως έχει ο κώδικάς μας (φτου φτου, μακριά από 'μάς).

Στα δεξιά της οθόνης και στην περιοχή 6 του screenshot βρίσκεται ο Inspector. Σε αυτόν μπορούμε να δούμε τα επιμέρους components που έχουν προστεθεί στο επιλεγμένο game object (στην εικόνα έχει ήδη επιλεχθεί η κάμερα, κάνοντας κλικ στο αντίστοιχο όνομα στο παράθυρο Hierarchy).

Τι στο καλό είναι όμως αυτό το "game object"; Ώρα για εξηγήσεις!

Στην Unity, κάθε αντικείμενο που υπάρχει στη σκηνή μας είναι ένα game object. Τα game objects από μόνα τους δεν κάνουν κάτι. Είναι απλά containers. Σε αυτά θα προσθέτουμε τα components, δηλαδή τα εξαρτήματα ή αλλιώς τα συστατικά, που χρειάζονται κάθε φορά, ώστε να ολοκληρώνουμε τη λειτουργικότητα του αντικειμένου που θέλουμε. Για παράδειγμα, να υπενθυμίσουμε ότι στην εικόνα βλέπουμε την κάμερα της σκηνής μας. Στον Inspector μπορούμε να διακρίνουμε ότι το game object της κάμερας περιλαμβάνει πέντε components:

- Ένα transform, που είναι υπεύθυνο για τη θέση (position), την περιστροφή (rotation) και την κλίμακα (scale) του αντικειμένου. Είναι αδύνατο να υπάρχει game object στη σκηνή μας και να μην έχει ένα component τέτοιου τύπου.
- Ένα component τύπου Camera, που περιλαμβάνει τις ιδιότητες της κάμερας και μας επιτρέπει να την παραμετροποιήσουμε.
- Άλλα τρία components, τύπων GUI Layer, Flare Layer και Audio Listener, που δεν επιτρέπουν παραμετροποίηση αλλά είναι απαραίτητα για τη μηχανή. Για παράδειγμα, το τελευταίο είναι υπεύθυνο για το τι και από πού θα ακούει ο παίκτης (μιας και κοιτάζει μέσω της κάμερας).

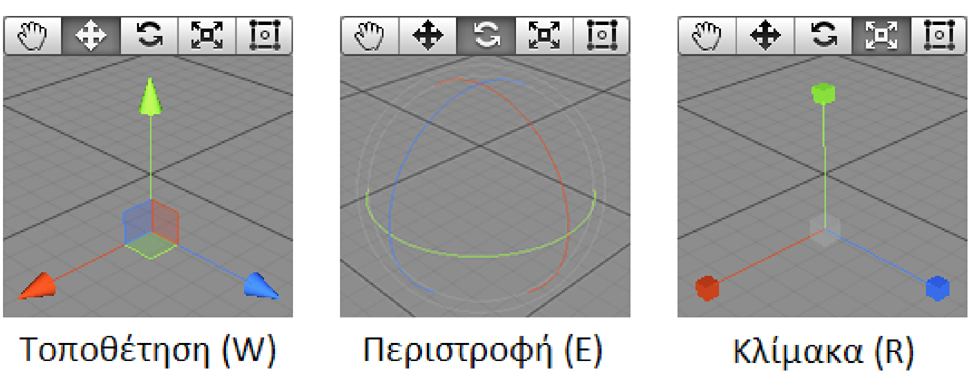
Κατά βάση, κάθε Component στη Unity είναι ένα script. Άλλα είναι ενσωματωμένα και μας τα παρέχει η μηχανή, άλλα έχουν δημιουργηθεί από εμάς. Στα επόμενα τεύχη θα μάθουμε πώς να φτιάχνουμε τα δικά μας scripts και να παρουσιάζουμε προς αλλαγή τις ιδιότητές τους.

Συνεχίζουμε στην περιοχή 3 του screenshot, όπου και παρατηρούμε τρία κουμπιά. Είναι υπεύθυνα για την εκτέλεση του παιχνιδιού, αλλά σε περιβάλλον μηχανής. Το πρώτο εκτελεί το παιχνίδι και φέρνει το focus στο παράθυρο "Game". Το δεύτερο παγώνει το παιχνίδι στο σημείο που είμαστε κατά την εκτέλεσή του, ενώ το τρίτο προχωράει το παιχνίδι κατά ένα frame, όταν αυτό είναι παγωμένο. Το τελευταίο βοηθάει πολύ σε περίπτωση που έχουμε να κάνουμε με physics.

Επί τη ευκαιρία, μια μικρή παρένθεση. Φανταστείτε τι θα γινόταν σε περίπτωση που, για να δοκιμάσουμε κάτι στο καινούργιο μας παιχνίδι, θα έπρεπε κάθε φορά που προσθέταμε κάτι, να χρειαζόταν να κάνουμε build το παιχνίδι. Όσο το παιχνίδι θα ήταν μικρό, δεν θα υπήρχε πρόβλημα. ‘Οταν όμως το μέγεθος θα άρχιζε να ξεφεύγει, άλλο τόσο θα ξέφευγε και η ώρα που θα χρειαζόταν η διαδικασία. Ένας εμπειρικός κανόνας είναι ότι αν χρειαστούμε πάνω από 5 δευτερόλεπτα να τεστάρουμε μια μικρή αλλαγή στο παιχνίδι, το testing γίνεται ανυπόφορο :/

Η Unity μάς επιτρέπει να τρέχουμε το παιχνίδι μας μέσα από τη μηχανή και να κάνουμε αλλαγές on the fly, ακόμη και όταν το παιχνίδι τρέχει. Χάρη σ' αυτή τη δυνατότητα μπορούμε να κάνουμε μικρές (ή μεγάλες) αλλαγές στις παραμέτρους του παιχνιδιού, χωρίς να ξοδεύουμε άπειρες ώρες κάνοντας compile και build. Προσοχή! Όταν το παιχνίδι τρέχει οι τιμές που αλλάζουμε στα αντικείμενα της σκηνής μας δεν αποθηκεύονται και δεν θα είναι διαθέσιμες όταν σταματήσουμε την εκτέλεσή του.

Στην αριστερή πάνω γωνία του παραθύρου στη Unity (στην περιοχή 1 της εικόνας), έχουμε πέντε στο πλήθος κουμπιά. Τα shortcuts των κουμπιών αυτών είναι τα πλήκτρα [Q], [W], [E], [R], [T] αντίστοιχα. Με επιλεγμένο το πρώτο κουμπί, μπορούμε να μετακινήσουμε το viewport (όχι την κάμερα) του παραθύρου της σκηνής μας, κάνοντας κλικ και σέρνοντας το ποντίκι μας. Κρατώντας πατημένο το [Alt] και σέρνοντας, μπορούμε να περιστρέψουμε το viewport στο σημείο που βρίσκεται, ενώ σέρνοντας με δεξί κλικ μπορούμε να κάνουμε zoom in/out. Το τελευταίο το επιτυγχάνουμε και με τη ροδέλα του ποντικιού.



Τα gizmos όπως εμφανίζονται στο παράθυρο της σκηνής, ανάλογα με την επιλογή που έχουμε κάνει (ή το πλήκτρο που έχουμε πατήσει).

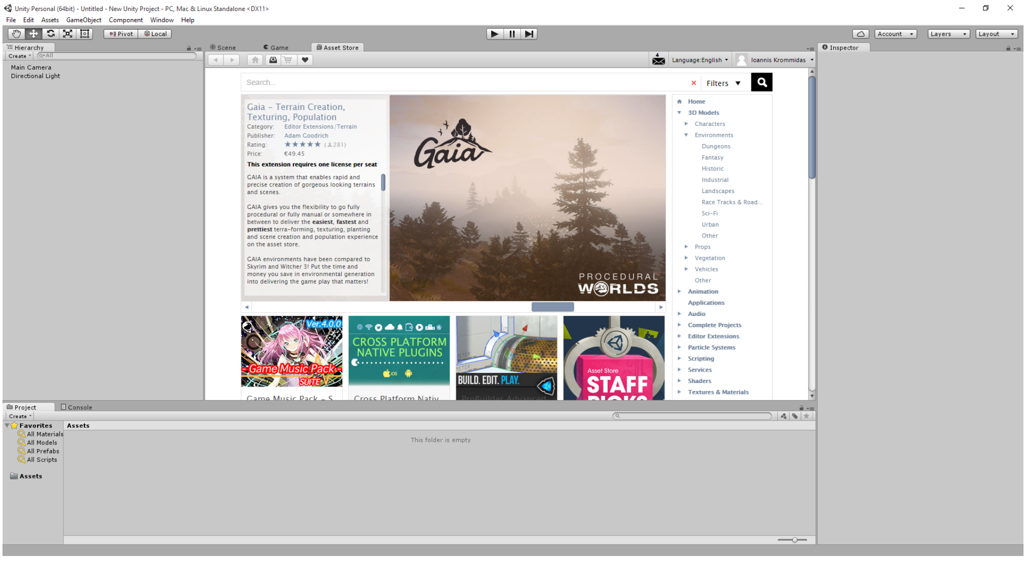
Με τα τέσσερα τελευταία αυτά κουμπιά μπορούμε να αλλάξουμε τις ιδιότητες του transform component, ενός game object. Το πρώτο κουμπί εμφανίζει το gizmo της τοποθέτησης. Με αυτό μπορούμε να μετακινήσουμε το αντικείμενό μας σε συγκεκριμένους άξονες (κάνοντας κλικ σε ένα και σέρνοντας το ποντίκι) ή σε ζευγάρι αυτών (π.χ., ταυτόχρονη μετακίνηση στους άξονες x και z). Το δεύτερο κουμπί εμφανίζει το gizmo της περιστροφής. Δουλεύει αντίστοιχα με το gizmo της τοποθέτησης. Το τρίτο είναι το gizmo της κλίμακας. Με αυτό μπορούμε να μεγεθύνουμε ή να συρρικνώσουμε τα αντικείμενά μας ως προς έναν ή όλους τους άξονες. Το τελευταίο κουμπί της ομάδας θα μας απασχολεί μόνον όταν έχουμε να κάνουμε με γραφικά 2D, οπότε ας το αφήσουμε προς το παρόν.

Τέλος, στην περιοχή 2 της εικόνας, βρίσκονται άλλα δύο κουμπιά. Αυτά έχουν πιο εξειδικευμένη χρήση. Επιγραμματικά να αναφέρουμε ότι το δεύτερο εναλλάσσει μεταξύ των τοπικών και των καθολικών αξόνων. Βοηθάει σε περίπτωση που θέλουμε να δούμε την περιστροφή ενός child object (πιο μέσα στην ιεραρχία της σκηνής) σε σχέση με τους καθολικούς άξονες (μιας και τα transforms των children ενός game object έχουν τη θέση τους, την περιστροφή τους και την κλίμακά τους σε σχέση με τις αντίστοιχες ιδιότητες του γονικού αντικειμένου). Το πρώτο τοποθετεί το σημείο περιστροφής του αντικειμένου είτε στη μέση (center) είτε στο πραγματικό του κέντρο (όπως αυτό ορίζεται από το πρόγραμμα που δημιουργήθηκε το μοντέλο, για παράδειγμα).

Πολύ χρήσιμο κουμπί στη Unity είναι το [F]. Αντιστοιχεί στη λειτουργία Frame Selected του μενού Edit. Όταν έχουμε επιλεγμένο ένα αντικείμενο στην ιεραρχία και έχουμε το ποντίκι μας στο παράθυρο της σκηνής, πατώντας το πλήκτρο [F] το viewport θα επικεντρωθεί και θα μας δείξει το επιλεγμένο game object. Αντίστοιχα, αν το ποντίκι βρίσκεται πάνω από το παράθυρο της ιεραρχίας, η λίστα θα σκρολάρει για να μας δείξει το επιλεγμένο αντικείμενο (σε περίπτωση που η λίστα είναι μεγάλη και δεν το βρίσκουμε εύκολα).

Ας μιλήσουμε λίγο για τα πακέτα που εγκαταστάθηκαν με τη μηχανή.

Τα πακέτα που έχουν συμπεριληφθεί περιλαμβάνουν γραφικά και λειτουργίες (κώδικα) που μπορούν να βοηθήσουν στο prototyping ενός παιχνιδιού. Για παράδειγμα, το πακέτο Characters περιλαμβάνει κώδικα και μοντέλα για τη μετακίνηση του χαρακτήρα του παίκτη στην πίστα. Η τρέχουσα έκδοση περιέχει δύο prefabs, ένα για κίνηση χαρακτήρα με κάμερα τρίτου προσώπου κι ένα για κίνηση χαρακτήρα με κάμερα πρώτου προσώπου. Άλλο ένα παράδειγμα αποτελεί το πακέτο Cameras. Περιλαμβάνει έτοιμες προς χρήση κοινές (που χρησιμοποιούνται από πολλά παιχνίδια) κάμερες. Τα πακέτα αυτά, πέρα από τον κώδικα και τα γραφικά (ή ήχους, εικόνες κ.λπ.) περιέχουν προκατασκευασμένα game objects, έτοιμα προς χρήση. Αυτά είναι γνωστά ως Prefabs. Τα asset packages μπορούν να εισαχθούν στα projects μας είτε από το asset store (θα το βρείτε στο μενού Window), αφού τα κατεβάσουμε, είτε από το μενού Assets → Import Package.



Το asset store είναι η μεγαλύτερη πηγή assets για τη Unity που υπάρχει στο Διαδίκτυο. Είναι προσπελάσιμη από browser (στο https://www.assetstore.unity3d.com) ή από παράθυρο στην ίδια τη μηχανή (Window → Asset Store).

Σημαντική έννοια στον κόσμο της Unity είναι το Prefab. Τα prefabs είναι στην ουσία assets, που περιλαμβάνουν game objects με τα components που τους έχουμε ορίσει μαζί με τις αρχικές τιμές αυτών. Μπορούμε να τα τοποθετήσουμε εύκολα στη σκηνή (και περισσότερα από ένα αν χρειαστεί), είτε σέρνοντάς τα από το παράθυρο Project είτε μέσω κώδικα. Για παράδειγμα, κομμάτια της πίστας (οι τοίχοι ενός σπιτιού, φράχτες, τα δέντρα κ.λπ.) ή οι εχθροί, θα είναι αποθηκευμένα ως prefabs ώστε να μπορούν να χρησιμοποιηθούν πολλές φορές χωρίς να χρειαστεί να επαναλάβουμε τη διαδικασία δημιουργίας τους από την αρχή. Το καλό με αυτά είναι ότι αν αλλάξουμε την τιμή μιας ιδιότητάς τους σε κάποιο prefab (επιλέγοντάς το στο παράθυρο project), η τιμή θα αλλάξει σε όλα τα υπάρχοντα instances στη σκηνή. Αν για κάποιο λόγο δεν το επιθυμούμε όμως, υπάρχει τρόπος να το απαγορεύσουμε.

Από νωρίς στη δράση

Αρκετή θεωρία μέχρι στιγμής. Ας πάρουμε μια γεύση του τι μπορεί να κάνει η μηχανή, δημιουργώντας μια μικρή πίστα με ένα χαρακτήρα που μπορεί να κινείται σ' αυτή.

1. Στο project που δημιουργήσαμε, στο μενού Assets → Import Packages εισάγουμε (Import) τα πακέτα Cameras, Characters και Prototyping.
2. Στο παράθυρο Hierarchy, ακολουθούμε τη διαδρομή Create → 3D Object → Terrain. Δημιουργούμε έτσι έδαφος για να περπατήσει ο χαρακτήρας μας.
3. Στο παράθυρο Project (πλέον, μπορούμε να δούμε ότι έχουν προστεθεί φάκελοι με assets) και στον φάκελο Assets/Standard Assets/Cameras/Prefabs, βλέπουμε τέσσερα αντικείμενα τα οποία είναι έτοιμα prefabs. Σέρνουμε το

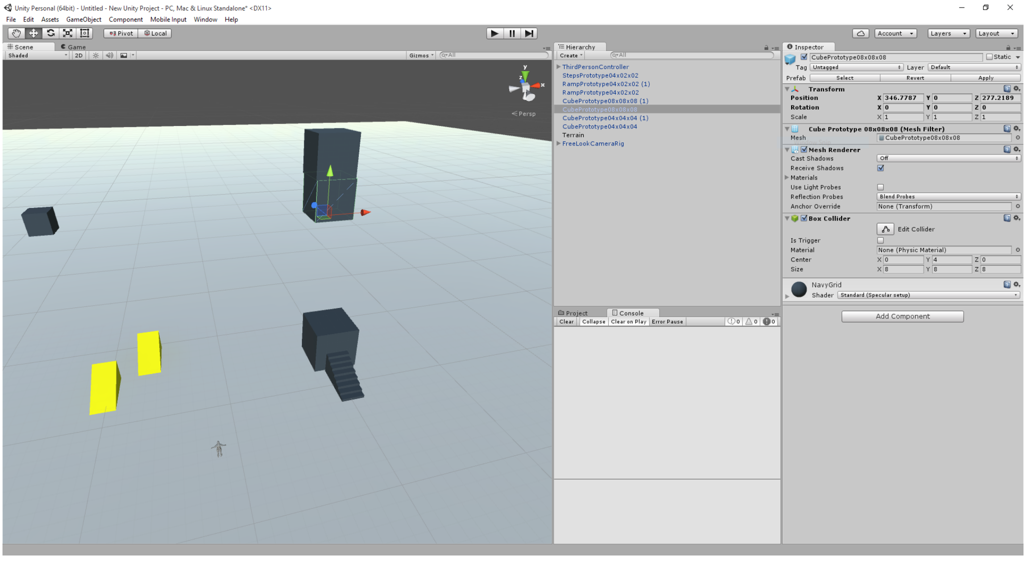
FreeLookCameraRig στο παράθυρο Hierarchy για να το προσθέσουμε στη σκηνή. Αυτή τη στιγμή, στη σκηνή υπάρχουν δύο κύριες κάμερες, πράγμα που θα κάνει τη Unity να φρικάρει όταν πάμε να "τρέξουμε" τη σκηνή.

4. Στο παράθυρο Hierarchy επιλέγουμε το game object με όνομα Main Camera και το διαγράφουμε με το Delete.
5. Στο παράθυρο Project της Unity, πηγαίνουμε στο φάκελο Assets/Standard Assets/Characters/ThirdPersonCharacter/Prefabs και προσθέτουμε στη σκηνή ένα ThirdPersonController prefab, σέρνοντάς το στο παράθυρο Hierarchy.
6. Σε περίπτωση που ο χαρακτήρας (ονομάζεται Ethan) είναι κάτω από το terrain (πατάμε [F] για να εστιάσει η κάμερα σε αυτόν), αν πατήσουμε Play θα δούμε ότι η βαρύτητα που υπάρχει στη σκηνή τον τραβάει κάτω. Για να λύσουμε το πρόβλημα, μέσω του transform gizmo (κουμπί [W]) τον μετακινούμε πάνω από το terrain που προσθέσαμε. Η μετακίνηση πρέπει να γίνει στη σκηνή και όχι στην ιεραρχία,αφού έχουμε σταματήσει την εκτέλεση του παιχνιδιού πατώντας Stop ή [Ctrl+P].
7. Για να υποχρεώσουμε την κάμερα να ακολουθεί τον παίκτη, πρέπει να δούμε τις ιδιότητές της στον Inspector. Την επιλέγουμε στο παράθυρο Hierarchy.
8. Στον Inspector βλέπουμε ότι η κάμερα αποτελείται από τέσσερα components. Αυτό που μας αφορά είναι το τρίτο κατά σειρά, ονόματι FreeLookCam. Η πρώτη ιδιότητά του ονομάζεται Target και δέχεται ως τιμή ένα αντικείμενο τύπου Transform. Σε αυτή την ιδιότητα πρέπει να ορίσουμε το αντικείμενο του game object του χαρακτήρα, που βάλαμε λίγο πριν στη σκηνή. Αυτό γίνεται με δύο τρόπους:
    - Ο πρώτος είναι να πατήσουμε στο κυκλάκι στα δεξιά της ιδιότητας. Αυτό θα εμφανίσει ένα παράθυρο στο οποίο μπορούμε να κάνουμε την επιλογή μας. Περιέχει δύο καρτέλες. Η καρτέλα Scene μάς δείχνει όλα τα αντικείμενα που μπορούμε να αναθέσουμε ως τιμή στην ιδιότητα και βρίσκονται στη σκηνή. Η καρτέλα Assets θα μας δείξει μια λίστα με όλα τα αντικείμενα που υπάρχουν στο project και μπορούμε ν' αναθέσουμε ως τιμή στην ιδιότητα. Το ότι δεν εμφανίζει τίποτα αυτή τη στιγμή οφείλεται στο ότι τα αντικείμενα δεν έχουν αρχικοποιηθεί ακόμα, δηλαδή δεν έχουν προστεθεί στη σκηνή. Από την καρτέλα Scene μπορούμε να επιλέξουμε το ThirdPersonController, με διπλό κλικ.
    - Ο άλλος και πιο εύκολος τρόπος είναι να σύρουμε το ThirdPersonController, από το παράθυρο Hierarchy στο πεδίο της ιδιότητας Target του script. Αυτό θα αναθέσει την αναφορά του αντικειμένου αυτόματα.
9. Πατώντας Play (ή [Ctrl+P]) για να τρέξουμε το παιχνίδι, πλέον θα δούμε ότι η κάμερα ακολουθεί τον Ethan και μπορούμε να τον κινήσουμε με τα [W], [A], [S], [D], [Space], [Ctrl] και [Shift].

Μια και η πίστα είναι λίγο άδεια, ας προσθέσουμε μερικά εμπόδια. Αυτά βρίσκονται στον φάκελο Assets/Standard Assets/Prototyping/Prefabs. Με την ίδια γνωστή διαδικασία του drag and drop προσθέτουμε αντικείμενα σε διάφορες θέσεις πάνω

στο terrain. Αν θελήσουμε να τοποθετήσουμε κάποιο αντικείμενο κατευθείαν σε ένα συγκεκριμένο σημείο στη σκηνή μας, μπορούμε αντί να το σύρουμε στο παράθυρο της ιεραρχίας, να το σύρουμε στο παράθυρο της σκηνής.

Ο Ethan δεν μπορεί να περάσει μέσα από τα εμπόδια (είναι ήδη σεταρισμένα, όντας prefabs) και σε αυτό βοηθάει η ενσωματωμένη μηχανή physics. Όπως προαναφέραμε, τα assets που έρχονται με τη Unity κάνουν πολύ καλά τη δουλειά τους όταν θέλουμε να δημιουργήσουμε το πρωτότυπο ενός παιχνιδιού.



Η "ολοκληρωμένη" πίστα μας.

Η μηχανή Unity και η λειτουργικότητά της είναι τεράστια και είναι δυστυχώς αδύνατον να καλυφθεί πλήρως στις λίγες σελίδες του παρόντος άθρου. Υπάρχει όμως ένα αναλυτικότατο –και τρομακτικά μεγάλο, θα λέγαμε– manual, στη διεύθυνση http://docs.unity3d.com/Manual/UnityManual.html. Είναι διαθέσιμο και offline, κατά την εγκατάσταση της μηχανής.

Υπάρχουν πολλές πτυχές της Unity που αξίζουν να εξερευνήσουμε. Θα ήταν όμως πολύ καλύτερα να τις ανακαλύψουμε στην πράξη, δημιουργώντας ένα παιχνίδι. Ξέρετε κάτι; Αυτό ακριβώς θα κάνουμε από το επόμενο τεύχος.


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

Skill: Intermediate
Tags: Twitter, bots, Raspberry Pi, Python, tweepy

Φτιάξτε το δικό σας TweetBot

Τα bots θα μπορούσαν να χαρακτηριστούν ως οι δεύτεροι σε δραστηριότητα χρήστες του Διαδικτύου – μετά τους ανθρώπους. Η συνεισφορά τους σε όλο αυτό το κατασκεύασμα που αποκαλούμε web, δεν είναι διόλου ευκαταφρόνητη. Μολαταύτα, λόγω της κακόβουλης χρήσης τους από επιτήδειους, η έννοια του bot έχει από καιρό στιγματιστεί. Κόντρα σ' αυτή την αντίληψη, στο παρόν άρθρο φτιάχνουμε ένα bot το οποίο με τίποτε δεν θα μπορεί να χρησιμοποιηθεί κακόβουλα.

του Πέτρου Κυλαδίτη

Για τους πέντε ή έξι αναγνώστες του περιοδικού που δεν γνωρίζουν τι είναι τα bots, να πούμε ότι πρόκειται για προγράμματα που υφίστανται σε συστήματα συνδεδεμένα στο Διαδίκτυο και που εκτελούν διάφορες λειτουργίες, μερικές φορές μιμούμενα τον τρόπο χρήσης του ιστού από τους ανθρώπους. Τα πιο γνωστά και χρήσιμα bots είναι τα λεγόμενα spiders, δηλαδή τα προγράμματα των μηχανών αναζήτησης (κι όχι μόνο) που διατρέχουν τον ιστό μεταβαίνοντας από link σε link, φτιάχνοντας έτσι ένα ευρετήριο των σελίδων που συναντούν στην πορεία τους.

Στο σημείο αυτό ίσως ήδη σκέφτεστε ότι τα bots αποτελούν ευλογία αλλά και κατάρα για το Διαδίκτυο. Αναλογιστείτε, π.χ., τις αυτοματοποιημένες επιθέσεις σε σελίδες login δικτυακών τόπων, τις αυτοματοποιημένες εγγραφές χρηστών σε fora για εκμετάλλευση από πλευράς spam bots κ.ο.κ.

Εμείς πάντως σίγουρα δεν θέλουμε να προκαλέσουμε αναστάτωση ή προβλήματα. Το αντίθετο μάλιστα: Σκοπός μας είναι να δείξουμε πώς είναι δυνατόν να αξιοποιούμε τα bots για ένα καλύτερο web. Φυσικά, ο καλύτερος τρόπος ώστε να σας πείσουμε για τις προθέσεις μας, είναι να κατασκευάσουμε το δικό μας, εντελώς καλοπροαίρετο, bot.

Το πουλάκι τσίου

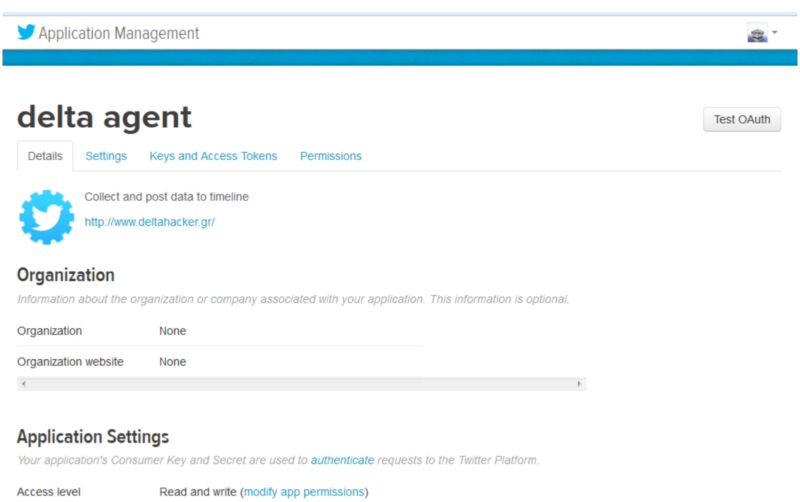
Τα bots δραστηριοποιούνται είτε γενικότερα στον ιστό, είτε ειδικότερα σε μια κατηγορία σελίδων (π.χ. fora), είτε ακόμα πιο ειδικά, σε ένα συγκεκριμένο δικτυακό τόπο. Το Twitter είναι ένας από αυτούς τους τόπους, στον οποίο δραστηριοποιούνται *και* bots. Βλέπετε, η δημοφιλής πλατφόρμα κοινωνικής δικτύωσης δεν διαθέτει μόνο ανθρώπους ως χρήστες, αλλά και μηχανές. Οι μηχανές (bots) που το χρησιμοποιούν, όπως ακριβώς και οι άνθρωποι, είτε διαβάζουν tweets είτε δημοσιεύουν νέα tweets. Όμως για ποιους λόγους να δημιουργήσει κάποιος ένα bot για το Twitter; Ο πιο απλός που μας έρχεται στο νου, είναι για να εκτελεί αυτοματοποιημένες δημοσιεύσεις.

Αυτές τις δημοσιεύσεις είτε θα τις έχει προετοιμάσει ο χρήστης του bot είτε θα δημιουργούνται αυτόματα, μετά από έλεγχο συνθηκών στο Internet ή/και στον πραγματικό κόσμο (ακούστηκε κάποιος να ψιθυρίζει "Internet of things" ή μου φάνηκε;)

Ένας τρόπος προκειμένου να φτιάξουμε το bot μας για το Twitter, είναι με την αξιοποίηση του Twitter API. Πρόσβαση στο API αποκτάμε αφού συνδεθούμε στο λογαριασμό μας στο Twitter και μεταβούμε στη διεύθυνση https://apps.Twitter.com. Εκεί, δημιουργούμε μια νέα εφαρμογή πατώντας στο κουμπί Create New App. Αμέσως παρουσιάζεται μια νέα σελίδα και μας ζητείται να δώσουμε πληροφορίες σχετικές με την εφαρμογή: το όνομά της, μια μικρή περιγραφή της λειτουργίας της, το URL του website απ' όπου οι ενδιαφερόμενοι θα μπορούν να την κατεβάζουν κ.ά.

Η εφαρμογή που τώρα φτιάχνουμε δεν προορίζεται για το ευρύ κοινό αλλά μόνον για εμάς. Έτσι, ως website μπορούμε να δώσουμε ό,τι θέλουμε. Εμείς, εντελώς τυχαία, ως site ορίσαμε το https://www.deltahacker.gr, ως τίτλο το "delta agent" κι ως περιγραφή δώσαμε το "Our post machine". Όπως καταλαβαίνετε, λίγη σημασία έχουν όλα αυτά. Η ουσία της υπόθεσης βρίσκεται στη σελίδα της εφαρμογής, η οποία είναι διαθέσιμη αφού τελειώσουμε με την καταχώρηση των προαναφερθέντων πληροφοριών.

Συγκεκριμένα, πηγαίνουμε στην τελευταία καρτέλα, με τίτλο Permissions, και σιγουρευόμαστε ότι είναι επιλεγμένο το Read and Write. Κάθε φορά που κάνουμε κάποια αλλαγή σ' αυτήν την καρτέλα, την επικυρώνουμε με κλικ στο κουμπί Update Settings. Αμέσως μετά μεταβαίνουμε στην καρτέλα Keys and Access Tokens. Σημειώνουμε τα αλφαριθμητικά που αντιστοιχούν στα Consumer Key και Consumer Secret. Πηγαίνουμε λίγο πιο κάτω κι επιλέγουμε τη δημιουργία Access Tokens πατώντας στο κουμπί Generate My Access Token and Token Secret. Σημειώνουμε τα νέα αλφαριθμητικά που δημιουργούνται. Τώρα, όσον αφορά στο κομμάτι της προετοιμασίας του Twitter, έχουμε τελειώσει. Σειρά έχουν πιο ευχάριστα και διασκεδαστικά πράγματα. Ναι, καλά το καταλάβατε, είναι ώρα να περάσουμε στο κομμάτι του προγραμματισμού.



Η δημιουργία της εφαρμογής μας μέσα από το λιτό περιβάλλον του Twitter, αποτελεί απλή διαδικασία. Την προσοχή μας αξίζουν οι πληροφορίες που βρίσκονται στην καρτέλα Keys and Access Tokens, καθώς είναι μέγιστης σημασίας για τη λειτουργία του bot.

Εν αρχή ην ο λόγος

Προκειμένου να συνδιαλεχθούμε προγραμματιστικά με το Twitter, αποφασίζουμε να χρησιμοποιήσουμε την αγαπημένη μας Python. Για καλή μας τύχη, για τη συγκεκριμένη γλώσσα υπάρχει μια βιβλιοθήκη, η tweepy, η οποία αναλαμβάνει όλη τη βρώμικη δουλειά της επικοινωνίας με το Twitter API, μέσα από ορισμένες βολικές συναρτήσεις. Στο δικτυακό τόπο της tweepy, στο http://www.tweepy.org, βρίσκουμε και την πλήρη τεκμηρίωση της πλούσιας αυτής βιβλιοθήκης.

Για την εγκατάστασή της υπάρχουν διάφορες εναλλακτικές. Η ευκολότερη είναι μέσω του συστήματος pip, η χρήση του οποίου περιγράφεται στο άρθρο του γράφοντα "Βγάλτε βόλτα την Python", στο τεύχος 034 του περιοδικού. Αν τώρα συγκαταλέγεστε στους 2-3 αναγνώστες από το Ανατολικό Τιμόρ, οι οποίοι δεν διάβασαν το εν λόγω τεύχος κι έτσι δεν έχουν το pip εγκατεστημένο, είναι ώρα να κάνετε κάτι επ' αυτού. Συγκεκριμένα, κατεβάστε το get-pip.py από το https://bootstrap.pypa.io/get-pip.py κι εκτελέστε το με τον διερμηνέα της Python. Στο σύστημά μας, ένα ηρωικό και καταταλαιπωρημένο Raspberry Pi, όλα αυτά τα πετύχαμε δίνοντας, διαδοχικά:

```
wget --no-check-certificate https://bootstrap.pypa.io/get-pip.py
python get-pip.py
```

Στη συνέχεια προχωρήσαμε στην εγκατάσταση της tweepy, εννοείται μέσω του pip:

```
pip install tweepy
```

Και ναι, καλά το καταλάβατε, το bot μας θα ζει και θ' αναπνέει μέσα σ' ένα Raspberry Pi. Ο λόγος είναι προφανής, καθώς επιθυμούμε να εκτελείται αδιάλειπτα, δίχως τη δική μας παρουσία, με τη μικρότερη δυνατή κατανάλωση ενέργειας. Οπότε τι καλύτερο από την τοποθέτησή του στο πάντοτε ανοικτό και σκληρά εργαζόμενο Pi μας, το οποίο μάλιστα μετράει αρκετούς μήνες uptime;

Στη σειρά σας, παρακαλώ!

Αφού καταλήξαμε στο περιβάλλον και στον τρόπο εργασίας, ας δούμε και τι ακριβώς θέλουμε να κάνει το bot. Για αρχή θα ασχοληθούμε με κάτι πιο απλοϊκό, όπως η αυτοματοποιημένη δημιουργία αναρτήσεων. Τα tweets, το bot θα τα διαβάζει από ένα αρχείο και θα τα αναρτά την ώρα που είναι ρυθμισμένο το κάθε ένα. Για να το πετύχουμε αυτό, στο αρχείο θα βρίσκονται τα tweets ένα ένα σε κάθε γραμμή, ενώ οι γραμμές θα είναι χωρισμένες σε δύο μέρη, μεταξύ των οποίων θα υπάρχει ένας χαρακτήρας tab. Στο πρώτο μέρος κάθε γραμμής θα βρίσκεται η ημερομηνία και ώρα που επιθυμούμε να γίνει το post, στη μορφή "εεεε,μμ,ηη,ωω,λλ" ("εεεε" είναι το έτος, "μμ" ο μήνας, "ηη" η μέρα, "ωω" η ώρα και "λλ" τα λεπτά). Στο δεύτερο τμήμα θα βρίσκεται το κείμενο του tweet. Το bot μας θα εκτελείται ανά τακτά χρονικά διαστήματα με έναν κανόνα που θα ορίσουμε ως cron job του συστήματος. Το μέγεθος του διαστήματος εξαρτάται από τις ανάγκες μας, δηλαδή από το πόσο συχνά και με

πόση χρονική ακρίβεια θέλουμε να ενεργοποιείται η διαδικασία της ανάρτησης των δημοσιεύσεών μας. Για μεγαλύτερη ακρίβεια μπορούμε να ζητάμε την εκτέλεση του script κάθε λεπτό, ενώ μια πιο φυσιολογική τιμή (για τις δικές μας ανάγκες, τέλος πάντων) πιστεύουμε ότι είναι η μία ώρα. Το bot μας θα εξετάζει τις γραμμές του αρχείου και θα συγκρίνει την τρέχουσα ώρα με την εκάστοτε προγραμματισμένη. Αν αυτή είναι ίδια ή έχει ξεπεραστεί, θα προβαίνει στην ανάρτηση του αντίστοιχου tweet. Στη συνέχεια θα αφαιρεί τις συγκεκριμένες γραμμές από το αρχείο και θα τερματίζει τη λειτουργία του μέχρι την επόμενη εκτέλεσή του. Ας τα δούμε όμως όλα αυτά στην πράξη, εξετάζοντας τον κώδικα που γράψαμε (οι αριθμοί γραμμών δεν υπάρχουν στον κώδικα αλλά έχουν προστεθεί εδώ για ευκολία στην αναφορά).

```
#!/usr/bin/python
from datetime import datetime
import tweepy
import time

consumer_key = "fgsbf53dafbafd35saf"
consumer_secret = "ghd43tftjuu9yygf6"
access_token = "26488189-5464yty554fdth554ser"
access_token_secret = "r534dsrgds4yhr5yds"

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

tweets_file_path = "/home/pi/tweets.txt"
tweets_file = open(tweets_file_path, "r")
tweets_list = tweets_file.readlines()
tweets_file.close()

tweets_new_list = ""

for tweet_line in tweets_list:
    tweet = tweet_line.split("\t", 1)
    tweet_time_list = tweet[0].split(",", 5)
    tweet_time = datetime(int(tweet_time_list[0]),
                          int(tweet_time_list[1]),
                          int(tweet_time_list[2]),
```

```
                              int(tweet_time_list[3]),
                              int(tweet_time_list[4]))
27     tweet_text = tweet[1]
28     now_time = datetime.now()
29     print (tweet_time - now_time).total_seconds()
30     if((tweet_time - now_time).total_seconds() <= 0):
31         time.sleep(10)
32         api.update_status(tweet_text)
33     else:
34         tweets_new_list = tweets_new_list + tweet_line
35
36 tweets_file = open(tweets_file_path, "w")
37 tweets_file.write(tweets_new_list)
38 tweets_file.close()
```

Τον κώδικα μπορείτε να κατεβάσετε από το http://bit.ly/dh052tbot1. Στη δεύτερη γραμμή εισάγουμε το αντικείμενο datetime από την ομώνυμη βιβλιοθήκη, το οποίο και θα μας χρειαστεί στη συνέχεια για τη σύγκριση της τρέχουσας ημερομηνίας με την επιθυμητή ημερομηνία ανάρτησης κάθε tweet. Στη συνέχεια εισάγουμε τη βιβλιοθήκη tweepy και, κλείνοντας με τις εισαγωγές, την time, η οποία με τη συνάρτηση sleep(s) θα μας βοηθήσει να ορίζουμε μια μικρή παύση ύστερα από κάθε ανάρτηση, ώστε αν ακολουθεί κι άλλη να υπάρχει ένα μικρό περιθώριο για τη διαχείρισή τους από πλευράς Twitter. Φυσικά, μπορούμε αν θέλουμε να ορίσουμε ακόμα μεγαλύτερο διάστημα – πάντα με βάση τις ανάγκες μας και τον σκοπό που θέλουμε να επιτύχουμε.

Στις γραμμές 06 έως και 09 έχουμε μια σειρά δηλώσεων μεταβλητών string, οι οποίες συγκρατούν τα νευραλγικά security keys και tokens της εφαρμογής αλλά και του λογαριασμού μας, μέσω των οποίων θα μας επιτραπούν η επικοινωνία με το timeline καθώς και η διαχείρισή του. Στη γραμμή 11 που ακολουθεί δημιουργούμε ένα αντικείμενο διαχείρισης της εφαρμογής, ορίζοντας τα consumer κλειδιά που έχουμε λάβει. Αμέσως μετά, στη γραμμή 12, ορίζουμε την πρόσβαση της εφαρμογής στο λογαριασμό μας, με τη βοήθεια των αντίστοιχων tokens. Χάρη σ' αυτό το αντικείμενο έχουμε πρόσβαση στο Twitter API, μέσω των πολύ φιλικών συναρτήσεων που παρέχει το tweepy.

Έχοντας τελειώσει με τη διασύνδεση του Twitter προχωράμε στην ανάγνωση των περιεχομένων του αρχείου στο οποίο έχουμε αποθηκεύσει τα tweets μας (βλ. γραμμές 16 έως και 19). Τις γραμμές του αρχείου τις αποθηκεύουμε σε μορφή λίστας, στη μεταβλητή tweets_list. Επίσης, στη γραμμή 21 ορίζουμε τη μεταβλητή tweets_new_list. Προσέξτε ότι, παρά το όνομά της, είναι μορφής string κι αποθηκεύει τις γραμμές που δεν αφορούν στα tweets που θ' αναρτηθούν κατ' αυτή την εκτέλεση. Τις γραμμές αυτές θα τις αποθηκεύσουμε λίγο πριν την έξοδο του προγράμματος, στις γραμμές

36 έως και 38, επικαλύπτοντας τα περιεχόμενα του αρχείου με τα tweets. Έτσι, κατά την επόμενη εκτέλεση του προγράμματος, οι γραμμές των tweets που έχουν αναρτηθεί θα έχουν αφαιρεθεί.

Συνεχίζοντας τώρα από τη γραμμή που είχαμε μείνει, την 23, ξεκινάμε ένα βρόγχο με τον οποίο εξετάζουμε γραμμή προς γραμμή τα προγραμματισμένα tweets. Την κάθε γραμμή τη σπάμε στα δύο, στο σημείο που υπάρχει ο χαρακτήρας tab. Το πρώτο κομμάτι που αφορά στην ημερομηνία και στην ώρα της ανάρτησης το σπάμε σε πέντε κομμάτια, τα οποία οριοθετούνται από το κόμμα. Με τη βοήθεια των πέντε αυτών στοιχείων της λίστας αρχικοποιούμε ένα αντικείμενο datetime. Όπως θα διαπιστώσετε, ο τρόπος που επιλέξαμε να αναπαραστήσουμε την επιθυμητή ημερομηνία ανάρτησης κάθε άλλο παρά τυχαίος είναι, αφού έτσι έχουμε πολύ εύκολα και με τη σειρά τις μεταβλητές που απαιτούνται για να δημιουργηθεί το αντικείμενο datetime, στο οποίο δείχνει η μεταβλητή tweet_time. Κατόπιν τούτου, αποθηκεύουμε το κείμενο του tweet, δηλαδή το δεύτερο τμήμα της αρχικής γραμμής, στη μεταβλητή tweet_text.

Στη γραμμή 32 σειρά έχει η δημιουργία ενός αντικειμένου datetime με την τρέχουσα ημερομηνία, στο οποίο δείχνει η μεταβλητή now_time. Συγκρίνοντας τη διαφορά τον δευτερολέπτων του tweet_time με του now_time (αξιοποώντας τη μέθοδο total_seconds()) μπορούμε να αντιληφθούμε αν η επιθυμητή ώρα εκτέλεσης του συγκεκριμένου post είναι τώρα (μηδενική ή αρνητική διαφορά) ή στο μέλλον. Αν ήλθε ή έχει έλθει η ώρα για ανάρτηση, τότε, αφού αφήσουμε ένα μικρό χρονικό διάστημα όπως ήδη περιγράψαμε, την εκτελούμε με τη βοήθεια της συνάρτησης update_status του αντικειμένου API. Αν πάλι η συγκεκριμένη γραμμή αφορά σε κάποια μελλοντική ανάρτηση, τότε, όπως βλέπετε στη γραμμή 34, την προσθέτουμε στο τέλος του κειμένου, το οποίο θα αποτελέσει το νέο περιεχόμενο του αρχείου των tweets, μετά το τέλος της εξέτασης όλων των γραμμών.

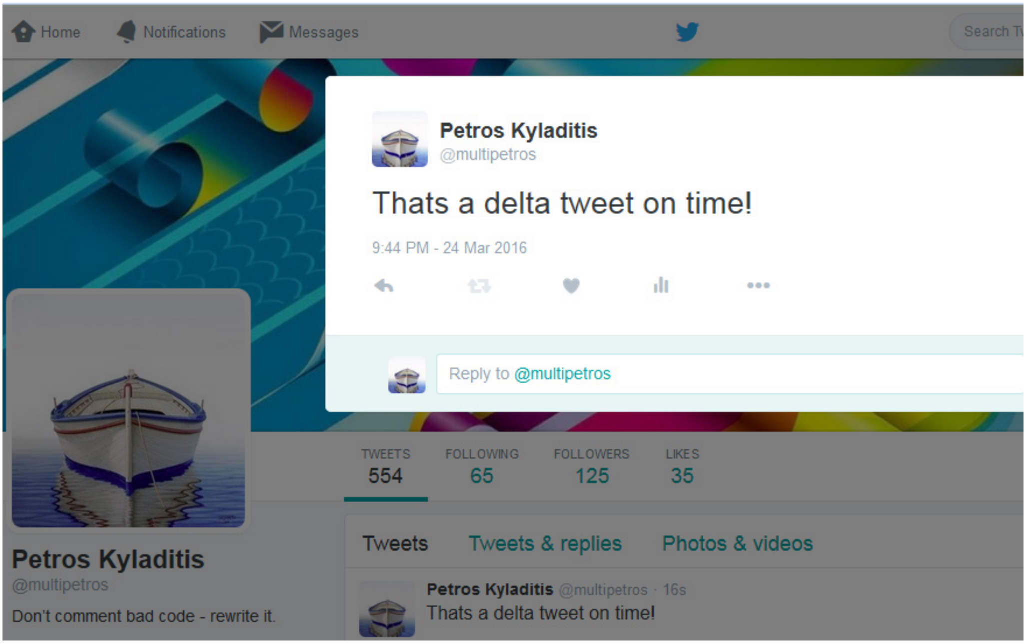
Αυτό λοιπόν ήταν όλο! Ούτε καν 40 γραμμές κώδικα και το bot μας είναι πανέτοιμο ν' αναλάβει δράση. Το μόνο που του λείπει είναι το έναυσμα, για το οποίο θα φροντίζει ευλαβικά ο δαίμονας του cron, σύμφωνα με τις οδηγίες που θα ορίσουμε στο crontab. Για να κάνουμε κάτι τέτοιο στο σύστημά μας, δίνουμε:

```
sudo crontab -e -u root
```

Στον editor που ανοίγει, προσθέτουμε τη γραμμή:

```
0,30 * * * * sudo python /project_dir/tweet_bot_scheduled.py
```

Αυτό θα κάνει το script μας να τρέχει με επαυξημένα δικαιώματα, κάθε μισή ώρα. Αν δε γνωρίζετε πολλά γύρω από το crontab και σας ενδιαφέρει να μάθετε καλά τη χρήση του, τότε μην παραλείψετε να διαβάσετε το πολύ επεξηγηματικό άρθρο στο τεύχος 016 του περιοδικού.



*Το προγραμματισμένο tweet εμφανίστηκε στην ώρα του. Τίποτα δεν προδίδει ότι υπεύθυνος για την ανάρτησή του *δεν* ήταν ο γράφοντας.*

Ένα Twitter bot με απ' όλα, παρακαλώ!

Το bot μας, αν και απλοϊκό δείχνει το δρόμο για πολλές κι ενδιαφέρουσες βελτιώσεις ή τροποποιήσεις. Αναλυτικότερα, έχουμε τη δυνατότητα είτε να επαυξήσουμε τις δυνατότητες του bot που ήδη διαθέτουμε, είτε να δημιουργήσουμε ένα νέο με εντελώς διαφορετική λογική. Για παράδειγμα, το νέο bot θα μπορούσε να συλλέγει στοιχεία από το web αλλά κι από το περιβάλλον του, να τα συνδυάζει και να παράγει το κατάλληλο αποτέλεσμα – πάντα στο timeline μας.

Αποφασίσαμε να υλοποιήσουμε μια εντελώς διαφορετική έκδοση του bot. Στη νέα του έκδοση, θα εκτελείται σε συγκεκριμένες ώρες και σε κάθε φορά θα υπάρχει ένας διαφορετικός (προκαθορισμένος) χαιρετισμός. Ταυτόχρονα, θα συλλέγει τη θερμοκρασία του SoC του Raspberry Pi, θα τραβάει ένα snapshot από μια web cam που έχουμε εγκαταστήσει (μπορεί η πρόσβαση να είναι ελεύθερη ή να χρειάζεται πιστοποίηση με όνομα χρήστη και κωδικό), και συνδυάζοντας όλα αυτά τα στοιχεία να δημιουργεί ένα όμορφο tweet.

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
from datetime import datetime
```

```
import tweepy
import os
#download image without auth
import urllib
#download image with auth
import urllib2
import base64

consumer_key = "fgsbf53dafbafd35saf"
consumer_secret = "ghd43tftjuu9yygf6"
access_token = "26488189-5464yty554fdth554ser"
access_token_secret = "r534dsrgds4yhr5yds"

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

cpu_temp = os.popen('vcgencmd measure_temp').readline()
cpu_temp = cpu_temp.replace('temp=','').replace('\n','')
img_path = "/home/pi/live-cam.jpg"

now_hour = datetime.now().hour
greetings = {7 : "Good morning",
13 : "Good afternoon",
17 : "Good evening",
23 : "Good night"}
if now_hour in greetings.keys():
    tweet_text = greetings[now_hour]
    tweet_text = tweet_text + " with a picture from beautiful Chios. My RPi's CPU temp right now is about " + cpu_temp
    download_img() #if no need auth
    #download_img_auth() #if need auth
    api.update_with_media(img_path, tweet_text)

```

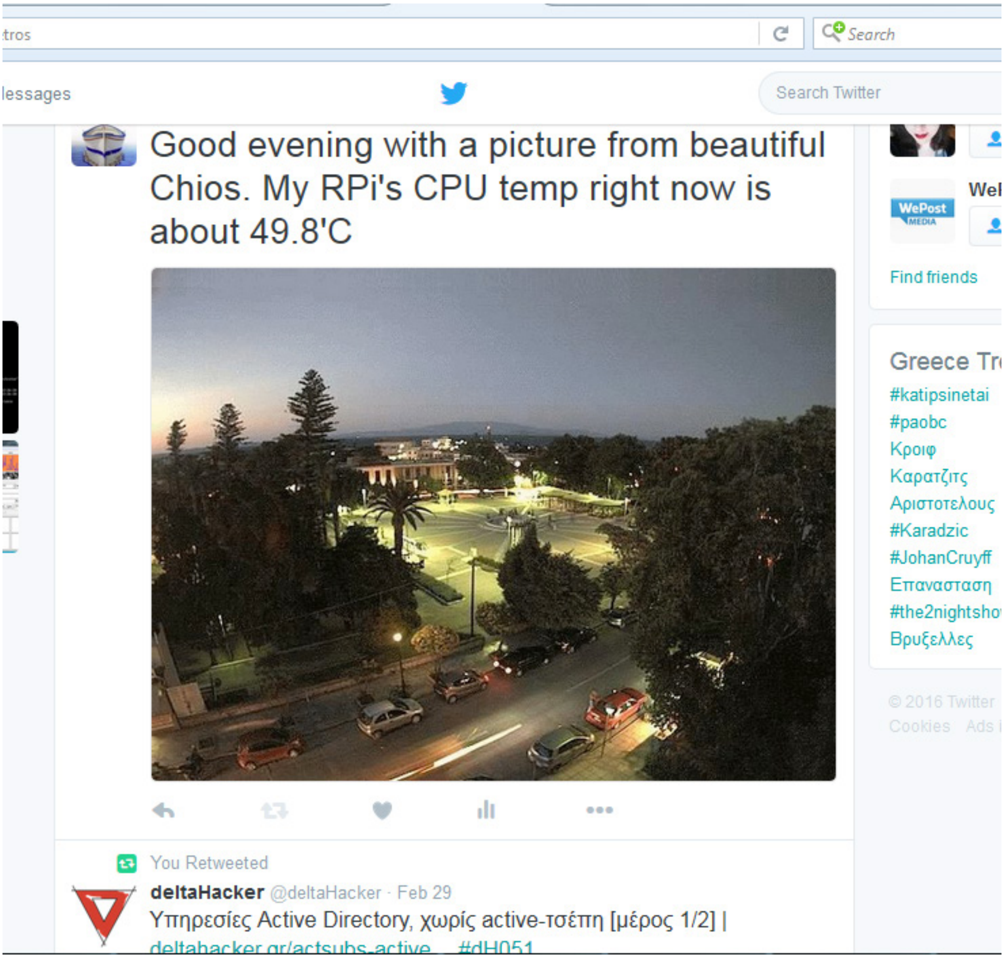
```
35 def download_img():
36     urllib.urlretrieve("http://10.20.30.81/cgi-bin/snapshot.cgi?channel=1", 
       img_path)
37
38 def download_img_auth():
39     request = urllib2.Request("http://10.20.30.81/cgi-bin/snapshot.
       cgi?channel=1")
40     base64string = base64.encodestring('%s:%s' % ("myusername", "mypass")).
       replace('\n', '')
41     request.add_header("Authorization", "Basic %s" % base64string)
42     result = urllib2.urlopen(request)
43     if result.headers.maintype == "image":
44          open(img_path, "wb").write(result.read())
```

Τον ανωτέρω κώδικα θα τον βρείτε στο http://bit.ly/dh052tbot2. Σε αυτή την έκδοχή του bot έχουμε αξιοποιήσει τη βιβλιοθήκη os, μέσω της οποίας εκτελούμε, στη γραμμή 22, την εντολή vcgencmd measure_temp, που μας επιστρέφει τη συμβολοσειρά με τη θερμοκρασία του SoC. Την εν λόγω συμβολοσειρά καθαρίζουμε από τους περιττούς χαρακτήρες στην αμέσως επόμενη γραμμή.

Εκτός της os, εισάγουμε και τις βιβλιοθήκες urllib, urllib2 και base64. Η συνάρτηση urlretrieve της πρώτης θα μας επιτρέψει να κατεβάσουμε την εικόνα στο Raspberry Pi μας από το URL της web κάμερας, στην περίπτωση που δεν απαιτείται ταυτοποίηση. Οι άλλες δύο θα μας επιτρέψουν να εκτελέσουμε ένα request προς τον server της web κάμερας, στον οποίο θα περάσουμε τα απαιτούμενα credentials κωδικοποιημένα κατά base64. Και για τους δύο αυτούς τρόπους έχουμε δημιουργήσει κατάλληλες συναρτήσεις, τις download_img και download_img_auth, που θα βρείτε στις γραμμές 35 έως 44.

Πίσω τώρα, στη γραμμή 26, λαμβάνουμε με άμεσο τρόπο την τρέχουσα ώρα με τη βοήθεια της μεθόδου hour, του αντικειμένου datetime. Αμέσως μετά, στη γραμμή 27, ορίζουμε ένα λεξικό, όπου τα κλειδιά του αντιστοιχούν στις επιθυμητές ώρες εμφάνισης του μηνύματος. Η συνέχεια έρχεται με έναν έλεγχο για να διαπιστώσουμε αν η τρέχουσα ώρα συνάδει με κάποια από αυτές για τις οποίες έχει οριστεί κάποιο μήνυμα στο λεξικό greetings. Αν όντως βρισκόμαστε σε κάποια απ' αυτές τις ώρες, συνδέουμε όλες τις συμβολοσειρές που αποτελούν το tweet μας και συλλαμβάνουμε μια εικόνα από τη web κάμερα (με τη βοήθεια της συνάρτησης που ταιριάζει στη μέθοδο πρόσβασης που υποστηρίζει). Στέλνουμε τέλος το tweet –με τη βοήθεια της μεθόδου update_with_media αυτή τη φορά–, περνώντας στο πρώτο όρισμα τη διαδρομή του αρχείου της εικόνας και στο δεύτερο το κείμενο που θα έχει το tweet.

Φυσικά, δεν πρέπει να λείψει και η κατάλληλη ρύθμιση του crontab, στο οποίο η εκτέλεση του script μπορεί να οριστεί είτε κάθε ώρα (μια και το ίδιο ελέγχει αν είναι ώρα να κελαηδήσει), είτε τις επιθυμητές ώρες και μόνο.



Ο συνδυασμός πληροφοριών από πλευράς Twitter bot, νομίζουμε πως έχει ιδιαίτερα εντυπωσιακά αποτελέσματα. Τι λέτε κι εσείς;